UNIVERSIDAD NACIONAL DE PIURA

FACULTAD DE INGENIERÍA INDUSTRIAL

ESCUELA PROFESIONAL DE INGENIERÍA INDUSTRIAL

PROYECTO DE TESIS

ALGORITMO EVOLUTIVO PARA ASIGNAR TRABAJOS A MÁQUINAS, CON DIFERENTE SECUENCIA DE OPERACIÓNES, DESDE LA PERSPECTIVA DE LA PLANIFICACIÓN DETALLADA.

PRESENTADA POR: Br. CYNTHIA TELLO OJEDA

Piura, 2014

TEL

UNIVERSIDAD NACIONAL DE PIURA

FACULTAD DE INGENIERÍA INDUSTRIAL

ESCUELA PROFESIONAL DE INGENIERÍA INDUSTRIAL

PROYECTO DE TESIS

ALGORITMO EVOLUTIVO PARA ASIGNAR TRABAJOS A MÁQUINAS, CON DIFERENTE SECUENCIA DE OPERACIÓNES, DESDE LA PERSPECTIVA DE LA PLANIFICACIÓN DETALLADA.

Br. CYNTHIA TELLO OJEDA

DR. ING. MIGUEL JIMÉNEZ CARRIÓN

Piura, 2014

UNIVERSIDAD NACIONAL DE PIURA

FACULTAD DE INGENIERÍA INDUSTRIAL

ESCUELA PROFESIONAL DE INGENIERÍA INDUSTRIAL

PROYECTO DE TESIS

ALGORITMO EVOLUTIVO PARA ASIGNAR TRABAJOS A MÁQUINAS, CON DIFERENTE SECUENCIA DE OPERACIÓNES, DESDE LA PERSPECTIVA DE LA PLANIFICACIÓN DETALLADA.

Dr. JULIO CÉSAR JIMÉNEZ CHAVESTA
PRESIDENTE – JURADO CALIFICADOR

Ing. TEOBALDO LEÓN GARCÍA, MSc.
VOCAL – JURADO CALIFICADOR

Ing. NÉSTOR MANUEL CASTILLO BURGOS
SECRETARIO – JURADO CALIFICADOR

Piura, 2014

ACTA DE SUSTENTACIÓN DE TESIS

Los Miembros del Jurado Calificador de la Tesis denominada: **«ALGORITMO EVOLUTIVO PARA ASIGNAR TRABAJOS A MÁQUINAS, CON DIFERENTE SECUENCIA DE OPERACIONES, DESDE LA PERSPECTIVA DE LA PLANIFICACIÓN DETALLADA»,** presentada por la señorita **CYNTHIA ELIANY TELLO OJEDA,** Bachiller de la Escuela Profesional en Ingeniería Industrial, asesorada por el **Dr. Miguel Jiménez Carrión;** reunidos para la sustentación de ésta y luego de escuchar su exposición y las respuestas a las preguntas formuladas, la declaran:

FACULTAD DE INGENIERÍA INDUSTRIAL Dirección de Investigación

Con el Calificativo: APROBADA

BUENO

En consecuencia la sustentante se encuentra **apta** para recibir el título profesional de **INGENIERO INDUSTRIAL**, conforme a Ley.

Piura,30 de diciembre del 2014

Dr. JULIO CÉSAR JIMÉNEZ CHAVESTA
PRESIDENTE – JURADO CALIFICADOR

Ing. TEOBALDO LEÓN GARCÍA, MSc.
VOCAL – JURADO CALIFICADOR

Ing. NÉSTOR MANUEL CASTILLO BURGOS
SECRETARIO – JURADO CALIFICADOR

DEDICATORIA

En primer lugar a Dios por habernos permitido llegar a este momento especial dándonos salud y fuerzas para lograr nuestros objetivos. En segundo lugar a nuestros padres por brindarnos su amor incondicional y por el apoyo para alcanzar nuestras metas profesionales y personales.

AGRADECIMIENTO

Durante este tiempo son muchas las personas que de una u otra manera han participado en este trabajo de realización de tesis y a quienes queremos expresar nuestra gratitud por el apoyo, ánimo, la confianza y sobre todo cariño y amistad que nos han prestado de forma desinteresada. Sin olvidar agradecer a la Universidad Nacional de Piura por ser parte importante en nuestra etapa de formación académica y profesional.

ESQUEMA DEL CONTENIDO

RESUMEN

En este trabajo se ha implementado un Algoritmo Genético Simple para resolver el problema de Job Shop Scheduling, conocido así dentro de lo que se llama planificación detallada de tareas; los resultados son altamente confiables tanto en la solución del problema como en el tiempo de ejecución del algoritmo, para problemas que tienen diferente secuencia de operaciones o la misma secuencia de operaciones, con tamaños de 15x15, y 15x20 y cuyo tiempos de ejecución de tareas son menores a 10 unidades de tiempo. Los parámetros del algoritmo recomendados para un buen funcionamiento son: tamaño de población 60 individuos y 80 generaciones. Así mismo se ha encontrado que los resultados están fuertemente influenciados por el diseño del cromosoma que no solo representa una solución factible sino también cumple con todas las restricciones, el diseñó de un procedimiento para determinar el makespan de cada solución y el diseño de la función de calidad para evaluar a los individuos, implementados.

Palabras Claves: Asignación de tareas, Algoritmo genético, Planificación detallada, Secuencia de operaciones

ABSTRACT

In this paper we have implemented a Simple Genetic Algorithm to solve the problem of Job Shop Scheduling and known within what is called detailed planning of tasks; the results are highly reliable both in solution and in the runtime of the algorithm, for problems that have different sequence of operations or the same sequence of operations, sizes 15x15 and 15x20 and whose execution times of tasks are less than 10 time units. The parameters of the recommended algorithm for smooth operation are: population size 60 individuals and 80 generations. Also it has been found that the results are strongly influenced by the chromosome design not only represents a feasible solution but also meets all constraints, the design of a procedure for determining the makespan of each solution and design function quality to evaluate individuals, implemented.

Keywords: Assigning tasks, Genetic algorithm, detailed planning, Sequence of operations

CAPÍTULO 1: INTRODUCCIÓN

Los problemas de planificación detallada de operaciones presentan las tareas que comprende la planificación tales como determinar la secuencia óptima de fabricación de artículos en un taller, la misma que es una tarea complicada debido al carácter combinatorio del problema, lo que dificulta la obtención de resultado óptimo para problemas muy grandes y tardaría cientos de años en encontrar una solución óptima.

Esto ha motivado un gran interés por este tipo de problemas. Dado que los problemas de planificación no son fáciles de resolver, muchos investigadores han desarrollado distintas metodologías y algoritmos que buscan mejorar la eficiencia de su solución. Es precisamente esta complejidad del problema lo que a motivado el desarrollo de este trabajo de investigación, en el que se planteó el uso de una heurística inspirada en los algoritmos evolutivos para abordar este problema conocido como JSSP (**J**ob **S**hop **S**cheduling **P**roblem); y encontrar una solución altamente satisfactoria en un tiempo reducido.

El JSSP consta básicamente de un conjunto de trabajos, donde cada uno tiene un conjunto de operaciones a ser procesadas en un conjunto de recursos, a los que se denomina máquinas. Dichas operaciones tienen un orden y un tiempo de procesamiento en cada una de las máquinas y éste no es modificable. Por tanto, el objetivo que se planteó en este trabajo es encontrar un plan de trabajo que cumpla con las restricciones del problema y que concluya todas las operaciones de los trabajos en el menor tiempo posible.

Los resultados en todas las instancias de prueba mostraron una alta eficiencia del algoritmo genético simple al encontrar el tiempo mínimo de ejecución de todos los trabajos con un tiempo de ejecución reducido.

1.1. DESCRIPCIÓN Y FORMULACIÓN DEL PROBLEMA

Se han desarrollado múltiples algoritmos para resolver el JSSP, pero debido a su complejidad en instancias grandes, no resulta posible contar con un método totalmente determinista para resolverlo en el caso general. De ahí que en

los últimos años, se haya desarrollado diversas heurísticas de solución del JSSP basadas en algoritmos genéticos, búsqueda tabú, recocido simulado y colonia de hormigas (Cortez Rivera, 2004).

Haciendo referencia a lo que dice Blazewicz, Ecker, Schmidt, & Weglarz, (1994), reportado por (Cortez Rivera, 2004), los problemas de planificación de tareas (scheduling) pueden ser descritos ampliamente como: "*el problema de acomodar los recursos en el tiempo para realizar un conjunto de tareas*". En la literatura de planificación de tareas se trata una gran variedad de problemas. En este trabajo se da una definición y una introducción del problema del Job Shop Scheduling. Por razones de brevedad y para hacer esta descripción concreta, se usan términos específicos de producción, pero el JSSP es importante no sólo en problemas de producción (por ejemplo el tránsito aéreo al despegue y aterrizaje o médicos y enfermeras cuidando a un paciente en un hospital, etc.).

Dentro de la gran variedad de problemas de planificación de recursos que existen, el JSSP es uno de los que han generado un mayor número de estudios. Esto se debe, sobre todo, a que es uno de los más difíciles de resolver. Además, el JSSP se utiliza en una amplia variedad de problemas del mundo real.

1.2. JUSTIFICACIÓN E IMPORTANCIA

Los modelos de optimización son modelos matemáticos que representan diferentes problemas y dependiendo del tipo de variables, y de la función objetivo a optimizar, estos pueden ser de programación lineal, de programación entera, de programación no lineal; problemas restringidos y problemas no restringidos; problemas que explotan en forma combinatoria como es el caso del problema de asignación de trabajos a máquinas conocido también como la planificación detallada de operaciones del tipo Job Shop.

Actualmente se cuenta con dos algoritmos conocidos como *Algoritmo óptimo de Johnson* y el *Algoritmo heurístico de Jackson* el primero aplica al problema de secuenciación de la producción continua en razón a que todos los trabajos siguen una misma secuencia, problema conocido como Flow Shop y el

segundo algoritmo se aplica a la producción intermitente en razón a que cada trabajo tiene una secuencia diferente. Sin embrago ambos resuelven el problema cuando el número de máquinas es 3 o menos.

En este proyecto se pretende dar una solución al problema planteado utilizando una de las herramientas inspiradas en la teoría de la evolución de los seres vivos, propuesta por Holland (1975) y reportado por García Martínez (2008), según éste Holland en su libro "Adaptación en sistemas naturales y artificiales", propone una clase de algoritmos adaptativos, ideas como una abstracción de la evolución para resolver problemas prácticos y para servir de modelos computacionales de sistemas naturales evolutivos, métodos que posteriormente recibieron el nombre de Algoritmos Genéticos (AGs).

Por tanto resolver el problema planteado utilizando un algoritmo evolutivo que resuelva el problema de producción continua y producción intermitente para más de tres máquinas beneficiará a las diferentes factorías y/o talleres del Perú, que enfrentan el problema de asignación de trabajos a máquinas conforme se ha indicado. Así mismo este algoritmo será el inicio para la construcción de un sistema informático basado en el algoritmo, que se adapte a cada factoría. De esta manera se justifica la investigación, al tratar de alcanzar los objetivos propuestos y dar respuesta a la siguiente pregunta:

¿Resuelve satisfactoriamente los Algoritmos Genéticos los problemas de planificación de las operaciones del tipo Job Shop?

1.3. OBJETIVO GENERAL Y ESPECÍFICOS

Objetivo General

Implementar un algoritmo genético simple para resolver el problema de asignación de trabajos a máquinas que tengan producción continua e intermitente, alcanzando el mínimo tiempo de ejecución de todos los trabajos, es decir determinar el makespan mínimo.

Objetivos Específicos

a) Analizar la estructura del modelo matemático del modelo de optimización a estudiar y diseñar su correspondiente cromosoma a ser implementado en el algoritmo genético.

b) Desarrollar el código e implementar el algoritmo genético computacionalmente para dar solución a diferentes instancias del problema planteado; evaluar el makespan y el tiempo de respuesta.

c) Planear un diseño de experimentos para determinar los parámetros de los algoritmos genéticos que mejor comportamiento tienen en la solución y el tiempo de respuesta.

1.4. HIPÓTESIS DE LA INVESTIGACIÓN

a) **Hp:** El diseño del cromosoma para el problema de planificación detallada de operaciones tipo Job Shop permite implementar el algoritmo genético y obtener soluciones altamente satisfactorias y en tiempo de respuesta aceptable.

b) **Ha:** El diseño del cromosoma para el problema de planificación detallada de operaciones tipo Job Shop no permite implementar el algoritmo genético y obtener soluciones altamente satisfactorias y en tiempo de respuesta aceptable.

CAPÍTULO 2: MARCO TEÓRICO DE LA INVESTIGACIÓN

2.1. DEFINICIÓN DEL PROBLEMA JOB SHOP SCHEDULING

El problema del Job Shop Scheduling, corresponde a un tipo de problema de planificación de tareas, y es un problema de optimización con dichas características; en donde uno de los objetivos principales es la disminución de tiempos en la tarea de planificación.

Consiste básicamente en planificar un conjunto de N trabajos $\{J_1,.., J_N\}$ sobre un conjunto de M recursos o máquinas físicas $(R_1,\ldots, R_M)$. Cada trabajo J_i consta de un conjunto de operaciones o tareas $\{\theta_{i1},\ldots,\theta_{iM}\}$ que deben ser ejecutadas de forma secuencial. Cada tarea θ_{ij} tiene asociado un tiempo de procesamiento sin interrupción de du_{ij} unidades de tiempo durante el cual requiere del uso exclusivo de un único recurso. Cada trabajo tiene un tiempo de inicio más temprano y en ocasiones se considera también un tiempo de término más tardío, lo que obliga a que los tiempos de inicio de las tareas tomen valores en dominios finitos. Cabe considerar que las operaciones de un mismo trabajo deben ejecutarse en un orden determinado, de forma que $\theta_{(i+1)j}$ no puede comenzar hasta que θ_{ij} haya terminado completamente. Además, las operaciones que comparten una misma máquina son no interrumpibles y mutuamente exclusivas. Las restricciones del problema se pueden resumir de la siguiente manera:

- Una tarea no puede visitar una misma máquina dos veces.
- No hay restricciones de precedencia entre operaciones de distintas tareas.
- Las operaciones no se pueden interrumpir.
- Cada máquina puede procesar sólo una tarea a la vez.
- No se especifican ni *release* times (fecha de tarea lista a ser procesada) ni "*due dates*" (fecha de entrega).

El objetivo consiste en encontrar una planificación factible, es decir, una asignación de tiempos de inicio st_{ij} para cada una de las tareas, que minimice

el *makespan* (*Cmax*) o tiempo de finalización de la última tarea. El Job shop se representa en la Figura N° 2.1, a través de un grafo conocido como grafo disyuntivo, el que se muestra con flechas llenas la duración de cada operación que se representa como un nodo al inicio de la flecha y además indica el nivel de precedencia entre una operación y otra, nodo terminal; así mismo se representa en flechas discontinuas las disyunciones adyacentes que existen en cada nodo.

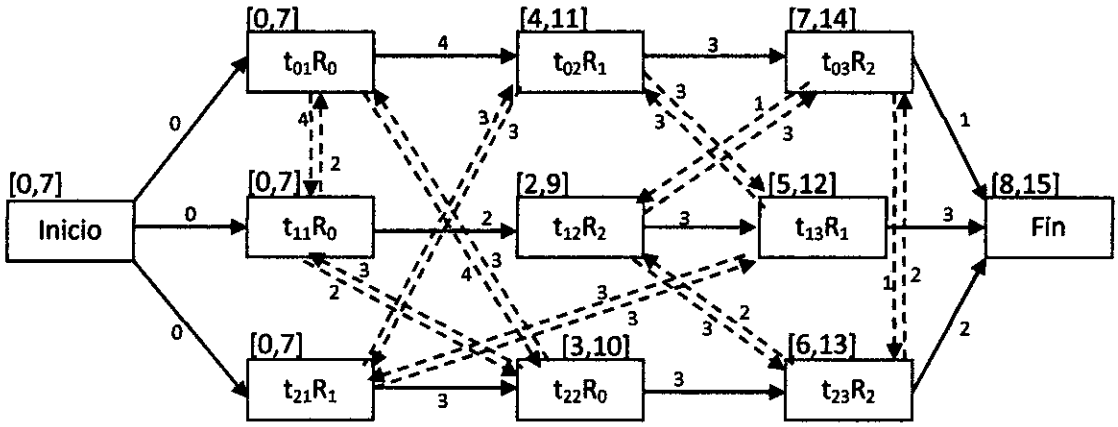


Figura N° 2.1: Representación gráfica de un problema con 3 trabajos de 3 tareas

El tiempo mínimo de inicio es 0 para todos ellos y el tiempo máximo de fin es 15. Los arcos están etiquetados con el costo de la tarea de origen

Las variantes para el problema del scheduling se relacionan con la secuencia de ordenación de las operaciones, denominada patrón de flujo. Si todos los trabajos tienen la misma secuencia de ordenación de las operaciones, el problema se conoce como *flow-shop*, mientras que si existe una ordenación determinada para cada uno de los trabajos y también hay un orden de uso de cada uno de los recursos por parte de las operaciones, el problema se denomina *permutation flow-shop*. Si no existe ninguna restricción de ordenamiento, el problema pasa a llamarse *open-shop*. El job-shop corresponde al problema en el cual cada trabajo tiene una ordenación determinada y se reitera es el problema que nos ocupa en este trabajo.

2.2. ESTADO DEL ARTE DEL PROBLEMA JOB SHOP

La investigación en teoría de scheduling ha venido desarrollándose en los últimos 50 años y ha sido objeto de mucha literatura con técnicas que van desde reglas de "despacho" (*dispatching*) no refinadas, hasta algoritmos

paralelos de ramificación y poda altamente sofisticados, heurísticas basadas en "cuellos de botella" (*bottleneck based heuristics*), y algoritmos genéticos paralelos. Además, dichas técnicas han sido formuladas desde un amplio espectro de investigadores, desde científicos de gestión, hasta expertos en producción. No obstante, con la aparición de nuevas metodologías, como redes neuronales y computación evolutiva, investigadores de campos como la biología, genética y neurofisiología han tenido también contribuciones regulares a la teoría de scheduling, poniendo de manifiesto la naturaleza multidisciplinaria de este campo (Peña & Zumelzu, 2006).

Diversas opiniones existen con respecto al origen del Job Shop Scheduling Problem (JSSP), conforme lo manifiesta Peña & Zumelzu (2006), exiten investigaciones que trataron con un problema consistente de *n* trabajos, *m* máquinas y relaciones de precedencia, por lo cual cada trabajo se procesa a través de las máquinas en un orden diferente.

Aunque no está claro quién debería llevarse el crédito por haber propuesto por primera vez el Job Shop Problem, se acepta que el libro "Industrial Scheduling" editado en 1964 por Muth y Thompson, constituye la base para la mayoría de las investigaciones que siguieron (Peña & Zumelzu, 2006).

Medina Duran, Pradenas Rojas, & Parada Daza, (2011), trabajando con Job Shop Flexible, proponen e implementa computacionalmente un algoritmo genético secuencial para resolver el problema del Job Shop Flexible, el cual es parte de la familia de los problemas de programación de tareas o trabajos (Scheduling) en un taller que funciona a pedido; sobre el particular manifiestan que el Job Shop flexible, surge como una generalización del problema del Job Shop y los resultados de su trabajo muestran la efectividad del algoritmo propuesto para entregar buenas soluciones en tiempos computacionales razonables en más de 130 instancias encontradas en la literatura.

Las técnicas de resolución del Job Shop Problem se pueden dividir en dos categorías; las técnicas de *optimización*, que producen una solución globalmente óptima, pero requiere un tiempo de computación muy alto, y las

técnicas de *aproximación*, que proporcionan una buena solución en un tiempo aceptable.

En general la eficiencia de los algoritmos que resuelven ciertos problemas complejos se mide por el tiempo de respuesta para encontrar la solución óptima, y por la capacidad de memoria que consume para hacerlo; en caso de ser el tiempo de respuesta muy alto buscar una respuesta satisfactoria en poco tiempo es normalmente la solución más adecuada, esto equivale a encontrar algoritmos con respuestas satisfactorias y tiempos de respuesta aceptables por quien debe tomar la decisión frente a la respuesta obtenida.

2.2.1 Técnicas de Optimización

En las técnicas de optimización, en general, la complejidad temporal crece exponencialmente con respecto al tamaño de la entrada. Los métodos obtenidos construyen una solución óptima a partir de los datos del problema siguiendo una serie de reglas que determinan de manera exacta el orden de procesamiento. Las principales técnicas de relacionadas con el trabajo de investigación son:

Programación lineal entera mixta

Propuesto inicialmente en 1960 por Alan Manne, reportado por (Peña & Zumelzu, 2006) este es un procedimiento muy utilizado en otros contextos de la programación matemática, sin embargo no resulta apropiado en este ámbito ya que su formulación lineal entera mixta da lugar a tiempos de ejecución elevados en problemas de tamaño real, lo cual la hace poco aconsejable.

Ramificación y poda (Branch & Bound)

Las técnicas de ramificación y poda hacen uso de una estructura de árbol construida dinámicamente, como forma de representar el espacio de todas las secuencias posibles. La búsqueda comienza en el nodo raíz y se continúa hasta llegar a un nodo hoja. Cada nodo en un nivel p de la búsqueda representa una secuencia parcial de p operaciones. Desde un

nodo no seleccionado, la operación de ramificación determina el siguiente conjunto de posibles nodos a partir del cual puede progresar la búsqueda. Las dos estrategias de ramificación más comunes son la Generación de Agendas Activas (Generating Active Schedules: GAS), y Ajuste de Conflictos Secuenciales (Settling Essential Conflicts: SEC). Cada nodo consiste en un secuenciamiento parcial de las operaciones sobre los recursos, pero mientras que la primera estrategia de ramificación fija el conjunto de operaciones siguientes a secuenciar, en la segunda se determina si una operación debería secuenciarse antes o después de otra.

El procedimiento de poda selecciona la operación con la que continuará la búsqueda y se basa en una estimación de una cota inferior y la mejor cota superior alcanzada hasta el momento. En los métodos de ramificación y poda es esencial afinar las cotas, ya que eso previene la necesidad de buscar en secciones grandes del espacio de soluciones.

2.2.2 Técnicas de Aproximación

Son las técnicas que proporcionan soluciones satisfactorias en tiempo de ejecución del algoritmo reducido, estas pueden ser heurísticas o metaheurísticas tales como:

Recocido simulado (*Simulated & Annealing*)

La denominación del algoritmo, Recocido Simulado, o Simulated Annealing, proviene de la estrecha analogía que guarda con el proceso del Recocido tal y como se usa en metalurgia. Éste consiste en que un metal fundido se va enfriando lentamente de manera que sus moléculas van adoptando poco a poco una configuración de mínima energía. Cuando comienza el proceso, a alta temperatura, las moléculas vibran y se desplazan caóticamente adoptando todo tipo de configuraciones en la estructura del metal de la que forman parte. A medida que la temperatura disminuye se va ralentizando el movimiento de las moléculas y éstas, de acuerdo con la Termodinámica, tienden a adoptar paulatinamente las configuraciones de menor energía, siendo ésta nula en el cero absoluto.

El Recocido Simulado intenta realizar numéricamente un proceso análogo al del recocido metalúrgico. El espacio de configuraciones no vendría ya dado por las posiciones de las moléculas, sino por los valores de una variable de interés, que en el caso del Job Shop será la secuencia de tareas que se desean procesar, mientras que el papel de la energía lo asumirá la función que se intenta minimizar, costo o tiempo de realización de tareas, por ejemplo.

Búsqueda tabú (Tabu Search)

Es una técnica de optimización iterativa global, que consiste en un procedimiento determinístico que restringe la búsqueda y evita los mínimos locales, almacenando la historia de búsqueda en memoria. Se prohíben movimientos entre vecinos que cumplan ciertas propiedades, con objeto de guiar el proceso de búsqueda para no duplicar soluciones previamente obtenidas. Una función de memoria a corto plazo permite "olvidos estratégicos" convirtiendo en "prohibido" los t movimientos más recientes. Sin embargo, el estado de un movimiento no es absoluto, ya que es posible seleccionar un movimiento tabú si alcanza un determinado nivel de calidad. También se dispone de funciones de memoria a largo plazo que pueden aplicarse para proporcionar una exploración más amplia del espacio de búsqueda. Finalmente existen estrategias a medio plazo o intermedias basadas en la modificación de las reglas de elección que favorecen la elección de movimientos y soluciones consideradas buenas históricamente, de forma que crean zonas de atracción del dominio de búsqueda e intensifican la búsqueda en dichas regiones. Los métodos a largo plazo diversifican la búsqueda en áreas no exploradas previamente.

El algoritmo de búsqueda tabú aplicada al Job Shop Problem genera vecinos mediante la inversión de arcos que unen tareas adyacentes en el camino crítico. Después de que un arco ($v \rightarrow w$) ha sido invertido, se introduce el paso inverso ($w \rightarrow v$) en la lista tabú, consistente en una lista de longitud definida que contiene los desplazamientos prohibidos. Cada vez que se realiza un número n de iteraciones, se modifica aleatoriamente

la longitud de la lista tabú. En cada iteración se evalúan todas las soluciones del vecindario que no se encuentran en la lista tabú, calculando para cada una el valor de C_{max} y seleccionándose la más prometedora.

Algoritmos genéticos (Genetic algorithms)

Los algoritmos genéticos imitan el procedimiento de la selección natural sobre el espacio de soluciones del problema considerado. Se basan en la creación de generaciones sucesivas de individuos representativos de posibles soluciones al problema. Los nuevos individuos se generan cruzando parejas seleccionadas dando mayor probabilidad a aquéllas soluciones que mejor valor de la función objetivo han obtenido en la generación anterior (individuos más adaptados al entorno).

En la literatura existen numerosas propuestas que resuelven el Job Shop Problem mediante el uso de Algoritmos Evolutivos. Bierwirth y Mattfeld, reportado por (Peña & Zumelzu, 2006), proponen varios Algoritmos Genéticos con distintas estrategias de evolución y un esquema de codificación denominado "permutaciones con repetición". De acuerdo con este esquema un cromosoma es en principio una permutación del conjunto de tareas en el que cada tarea viene representada simplemente por el número del trabajo al que pertenece. Por ejemplo, si tenemos un problema con 3 trabajos de 3 tareas cada uno, un cromosoma puede ser el siguiente (1 3 2 2 3 1 1 3 2). En esta codificación el primer 1 representa la primera tarea del primer trabajo, el segundo 1 la segunda y así sucesivamente. De este modo las tareas de cada trabajo aparecen representadas en el orden de ejecución secuencial, mientras que las tareas que requieren al mismo recurso pueden aparecer en cualquier orden. La ventaja principal que presenta este esquema de codificación es que permite diseñar operadores genéticos eficientes, de cruce y mutación, que producen cromosomas factibles.

Reglas de prioridad de despacho (Priority dispatch rules)

Las reglas de prioridad son probablemente las reglas heurísticas aplicadas con mayor frecuencia para resolver problemas de scheduling, esto se debe a su fácil implementación y a su baja complejidad de tiempo.

Los algoritmos de Giffler y Thompson, reportado por (Peña & Zumelzu, 2006) se pueden considerar como la base de todas las heurísticas basadas en reglas de prioridad. Éstos han propuesto dos algoritmos para generar schedules: procedimientos de generación de schedules activos y schedules nondelay. Un schedule nondelay tiene la propiedad de que ninguna máquina permanece ociosa si hay un trabajo disponible para su procesamiento. Un schedule activo tiene la propiedad de que ninguna operación se puede iniciar tempranamente sin retrasar otro trabajo.

Los schedules activos forman un conjunto más grande que incluye como subconjunto a los schedules nondelay. Así mismo el procedimiento de generación explora el espacio de búsqueda por medio de una estructura de árbol. Los nodos en el árbol representan los schedules parciales, los arcos representan las posibles elecciones, y las hojas del árbol son el conjunto de schedules. Para un schedule parcial, el algoritmo esencialmente identifica todos los conflictos de procesamiento, es decir, identifica operaciones que compiten por la misma máquina, y en cada etapa usa un procedimiento de enumeración para resolver esos conflictos bajo todas las formas posibles. En contraste, las heurísticas resuelven esos conflictos con reglas de prioridad de despacho; las heurísticas especifican una regla de prioridad para seleccionar una operación entre las operaciones en conflicto.

Heurísticas basadas en cuellos de botella (Bottleneck based heuristics)

Si bien durante bastante tiempo los únicos métodos de aproximación viables han sido las reglas de prioridad de despacho, recientemente con la llegada de computadores más potentes, así como en el énfasis puesto en nuevas técnicas cuidadosamente analizadas, diseñadas

e implementadas, ha dado lugar a aproximaciones más sofisticadas, llamadas *heurísticas basadas en cuellos de botella.*

Un ejemplo de esta aproximación es el *Shifting bottleneck procedure* (SBP). Un SBP se caracteriza por las siguientes tareas: identificación del subproblema, selección del cuello de botella, solución del subproblema, y reoptimización del scheduling. La estrategia real implica relajar el problema de scheduling $n \times m$, en m problemas de 1 máquina y resolver cada subproblema cada vez de forma iterativa usando la aproximación de Carlier. Cada una de las soluciones se compara con las demás y se ordenan las máquinas según su solución. La máquina no secuenciada que tiene la solución mayor se identifica como la máquina que provoca el cuello de botella. SPB secuencia dicha máquina basándose en las que ya han sido calculadas, ignorando el resto de máquinas no secuenciadas. La selección de la máquina que constituye el cuello de botella viene motivada por la conjetura de que el scheduling en etapas posteriores podría deteriorar el tiempo de realización del scheduling (makespan).

Cada vez que una máquina se identifica como de cuello de botella, todas las máquinas que ya han sido secuenciadas, y que son susceptibles de mejoras, se reoptimizan localmente resolviendo el problema de una máquina de nuevo. La principal contribución de esta aproximación es la forma en que se usa la relajación de una máquina para decidir el orden en el que las máquinas deben ser secuenciadas.

Trabajando con el JSSP Cortez Rivera (2004), propuso un algoritmo genético que utiliza una codificación basada en permutaciones con repeticiones. Esta es una representación muy sencilla y compacta, para la cual se diseñó un operador de mutación ad-hoc. Los resultados obtenidos indican que el algoritmo propuesto en esta tesis es una alternativa viable para resolver el JSSP, dada la calidad de las soluciones obtenidas, así como su costo computacional respectivo.

En este trabajo se desarrolla al detalle la técnica de los algoritmos genéticos para resolver el problema en cuestión, el cual se presenta más adelante.

2.2.3 El problema del Job Shop Scheduling Clásico

El JSSP tiene la característica de que los trabajos son unidireccionales. Otra característica importante es que la máquina que inicia no lo hace precisamente con el primer trabajo, sino que cualquier máquina puede empezar o terminar, ya que el orden de procesamiento de los trabajos en las máquinas es una característica de cada uno de los problemas. En la figura N° 2.1 se puede ver la solución de un problema que tiene tres trabajos y tres máquinas, esta solución se conoce como Plan de Trabajo para el JSSP, y es equivalente a un diagrama de Gantt en el cual el tiempo máximo representa el makespan.

2.2.4 Definición del problema

Hay m máquinas donde tienen que ser procesados n trabajos con m operaciones por trabajo. Cada operación tiene que ser procesada en una máquina. El orden de procesamiento de cada una de las operaciones de cada trabajo está dado por el problema y éste puede ser variable de un trabajo a otro, además una vez definido es inamovible. Una vez que una operación comienza a ser procesada no puede interrumpirse. Finalmente, todas las operaciones de todos los trabajos tienen la misma prioridad. Para poder ser más claro se listan las restricciones del JSSP clásico:

- No está permitido que dos operaciones del mismo trabajo se procesen simultáneamente.
- Ninguna operación tiene prioridad sobre las demás.
- Ningún trabajo puede ser procesado más de una vez en la misma máquina.

- Cada trabajo es procesado hasta concluirse, aunque haya que esperar y retardarse entre las operaciones procesadas.
- Un trabajo puede iniciarse en cualquier momento siempre y cuando esté disponible la máquina y no se haya especificado un tiempo de inicio.
- Los trabajos tienen que esperar a que la siguiente máquina esté disponible para que éste sea procesado.
- Ninguna máquina puede procesar más de un trabajo a la vez.
- Los tiempos de configuración y cambio de máquina son independientes del orden de procesamiento y éste está incluido en los tiempos de procesamiento.
- Hay sólo una máquina de cada tipo.
- Las máquinas pueden estar ociosas en cualquier momento del plan de trabajo.
- Las máquinas están disponibles en cualquier momento.
- Las restricciones tecnológicas están bien definidas y son previamente conocidas, además de que son inamovibles.

El objetivo a ser optimizado más usado en el JSSP es el de encontrar un plan de trabajo válido (que no viole ninguna de las restricciones antes mencionadas) que complete todos los trabajos en el menor tiempo posible. Este objetivo es conocido como *makespan* y es el que se minimiza. Los planes de trabajo factibles pueden obtenerse permutando el orden de las operaciones en las máquinas, teniendo cuidado de no violar las restricciones del problema. De acuerdo con esto, lo que se hace es una minimización a un problema de optimización combinatoria, ya que la representación de las soluciones son permutaciones sobre las operaciones de cada trabajo. Más específicamente, las operaciones a ser procesadas en una máquina forman la *secuencia de operaciones* para esa máquina. El plan de trabajo está formado de n secuencias de operaciones para cada máquina.

Puesto que cada secuencia de operaciones puede ser permutada independientemente de la secuencia de operaciones de otra máquina, el número total de posibles soluciones para el JSSP es $(n!)^m$, donde n denota el número de trabajos y m el número de máquinas. Este número constituye el espacio de búsqueda del problema. El siguiente ejemplo muestra el tamaño del problema para resolverse por métodos clásicos de búsqueda o enumeración.

Suponiendo que tenemos un problema de 24 operaciones y 1 máquina, el espacio de búsqueda para este problema es (24!) = 6.20×10^{23} posibles soluciones. Ahora bien, si presuponemos que la edad del universo es 20 mil millones de años tenemos que 20×10^9 x 365 x 24 x 60 x 60 = 6.31×10^{17} segundos, que es aproximadamente 6.31×10^{23} μseg. Si tuviéramos una computadora capaz de procesar una solución cada μseg, aun considerando toda la edad del universo apenas hubiéramos podido resolver este problema por enumeración. Esto nos da una idea de la complejidad del problema y esta es la razón principal por la que ha resultado de gran interés para los investigadores.

Jonhson y Garey (1979), reportado por (Cortez Rivera, 2004), demostraron que el JSSP es un problema NP-duro, y de entre esa clase es uno de los menos tratables.

Estandarizando la notación del JSSP

El JSSP que interesó resolver es de tamaño $m_x n$ y consiste de n trabajos $J_1, J_2, \ldots, J_n$ y m máquinas $M_1, M_2, \ldots, M_m$. Para cada trabajo J_i, se tiene una secuencia de k_i operaciones $O_i = o_{i1}, o_{i2}, \ldots, O_{ik_i}$ que describe el *orden de procesamiento* de las operaciones del trabajo J_i. El orden de procesamiento de los trabajos es algunas veces llamado *restricciones tecnológicas*. La operación o_{ij} es la *j-ésima* operación del trabajo i, y éste es procesado en una máquina específica Mo_{ij} y tiene un tiempo de procesamiento To_{ij}. El conjunto de operaciones P es denotado por T. Es importante mencionar que para este problema el conjunto de k_i operaciones es de tamaño m. Además el conjunto de operaciones O_i está

representado con los trabajos J_i, donde la primera aparición de J_i corresponde a o_{i1}, la segunda aparición a o_{i2} y así sucesivamente.

Cada trabajo puede tener asociado un tiempo de liberación (*release time*) r_i antes de que otro trabajo pueda procesarse y cada máquina un tiempo de preparación (*setup time*) u_i que se requiere antes de que se pueda hacer algo en la máquina. Es común trabajar con intervalos discretos de tiempo. La idea es que los tiempos de procesamiento, los tiempos de preparación y los tiempos de liberación sean enteros; sin embargo en este trabajo se considera que tanto el tiempo de preparación y el tiempo de liberación están incluidos en el tiempo de procesamiento. Cuando los trabajos son procesados en las máquinas, cada máquina puede procesar sólo un trabajo a la vez. Sólo una operación de cada trabajo puede ser procesada a la misma vez y ninguna operación tiene más prioridad que otra; es decir todas tienen la misma prioridad (esto es, que una vez que una operación es iniciada, no puede ser detenida hasta que ésta haya concluido).

Una instancia del JSSP se define mediante una matriz que contiene el orden de procesamiento de cada una de las operaciones, véase el cuadro N° 2.1, donde se muestra cada uno de los trabajos y el orden de cada una de las operaciones en las máquinas (a), y otra matriz donde se muestra el tiempo de procesamiento de cada una de las operaciones sobre cada una de las máquinas (b). Para facilitar su comprensión se muestra en un solo cuadro el orden de las máquinas en las que se va a procesar cada una de las operaciones del trabajo y el tiempo de procesamiento correspondiente a cada operación en cada máquina, véase el cuadro No 2.2

Un plan de trabajo es aquel donde se tienen los trabajos acomodados en las máquinas de tal forma que cumplan todas las restricciones del problema. Un plan de trabajo es generalmente visualizado mediante un diagrama de Gantt; un ejemplo se muestra en la figura 1.1. Cada renglón en el diagrama representa una máquina y cada cuadro representa una operación. Los cuadros están marcados con el número de

trabajo al que corresponde, es decir se identifican por el color y el número especificado. El tiempo que tarda en ejecutarse cada operación se especifica en el eje horizontal.

Cuadro N° 2.1: Instancia del JSSP

(a) Flujo de los trabajos

Trabajo	máquina		
0	0	1	2
1	0	2	1
2	1	0	2

(b) Tiempo de los trabajos

Trabajo	Máquina		
0	3	3	3
1	2	3	4
2	3	2	1

Cuadro N° 2.2: Flujo y tiempos de los trabajos en el JSSP

Trabajo	Máquina (tiempo)		
0	0(3)	1(3)	2(3)
1	0(2)	2(3)	1(4)
2	1(3)	0(2)	2(1)

Además, en el JSSP clásico, usualmente se requiere que para cada trabajo *Ji*, la secuencia de operaciones *Oi* contenga exactamente una operación para ser procesada por cada una de las máquinas del problema.

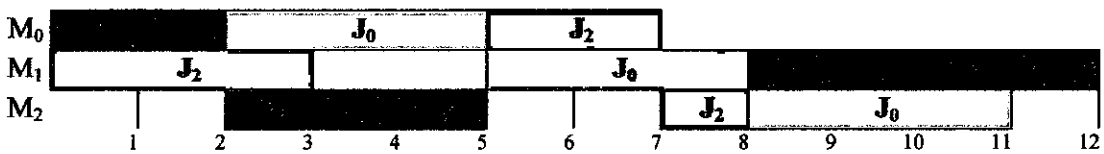


Figura No 2.2: Plan de trabajo, esto es, una solución para el JSSP con los datos del cuadro 2.2

Hay muchos objetivos a tratar en el JSSP, pero este trabajo de investigación sólo se refiere al cálculo del makespan es decir a determinar el tiempo mínimo de ejecución de todos los trabajos. Por tal motivo se amplía la notación para fines ilustrativos:

- C_i. El tiempo de terminación (completion time) de la última operación del trabajo J_i.

- F_i. El tiempo de flujo (flow time) del trabajo J_i.
- *di*. Algunos problemas especifican el tiempo comprometido d_i (due date), o sea el tiempo máximo para J_i, que es el tiempo en que a lo más debe ser concluido un trabajo.
- L_i. El retraso (*lateness*) de J_i, $L_i = C_i - d_i$. La medición del retraso es el tiempo que excede un trabajo en ser concluido sobre el tiempo comprometido. Si el trabajo termina antes, se dice que el retraso es negativo.
- T_i. La tardanza (*tardiness*) de J_i, $T_i = máx\ (L_i, 0)$.
- E_i. La puntualidad (*earliness*) de J_i, $E_i = máx\ (-L_i, 0)$.

Normalmente las medidas de desempeño para el JSSP son las siguientes:

makespan: $C_{máx} = máx_{i\epsilon\{1...n\}} C_i$. Tiempo mínimo para completar todos los trabajos.

total flow time: $F_\sum = \sum_{i\,\epsilon\,\{1...n\}} F_i$. Es el tiempo consumido por todos los trabajos.

total lateness: $L_\sum = \sum_{i\,\epsilon\,\{1...n\}} L_i$. La suma de todos los retrasos de los trabajos.

total tardiness: $T_\sum = \sum_{i\,\epsilon\,\{1...n\}} T_i$. La suma de todas las tardanzas de los trabajos.

total earliness: $E_\sum = \sum_{i\,\epsilon\,\{1...n\}} E_i$. La suma de todos los tiempos de terminación previstos al tiempo comprometido.

maximum lateness: $L_{máx} = máx_{i\epsilon\{1...n\}} L_i$. El retraso del más atrasado de los trabajos.

maximum tardiness: $T_{máx} = máx_{i\epsilon\{1...n\}} T_i$. La tardanza del más tardado de los trabajos.

Todas estas medidas de desempeño del JSSP se tienen que minimizar. Todas estas mediciones son importantes para muchas áreas, ya que lo más importante es reducir los costos en todas las formas, pero se reitera que para el caso específico en este trabajo, sólo se refiere a minimizar el tiempo para completar todos los trabajos (*makespan*).

2.2.5 Tipos de planes de trabajo

Los planes de trabajo generalmente se categorizan en las siguientes clases:

- Un plan de trabajo en que ninguna operación puede iniciarse sin cambiar el orden de procesamiento o sin que se viole alguna de las restricciones tecnológicas, es llamado *semiactivo*.
- Un plan de trabajo en que ninguna operación puede iniciarse sin que se retarde cualquier otra operación o sin que se viole alguna de las restricciones tecnológicas, es llamado *activo*.

En la Figura N° 2.3, se puede observar que en la primera gráfica, se muestra un plan de trabajo semiactivo, en donde ve que se puede cambiar la tercera operación del trabajo J_2, que si se inicia antes no afecta el tiempo de operación del plan de trabajo.

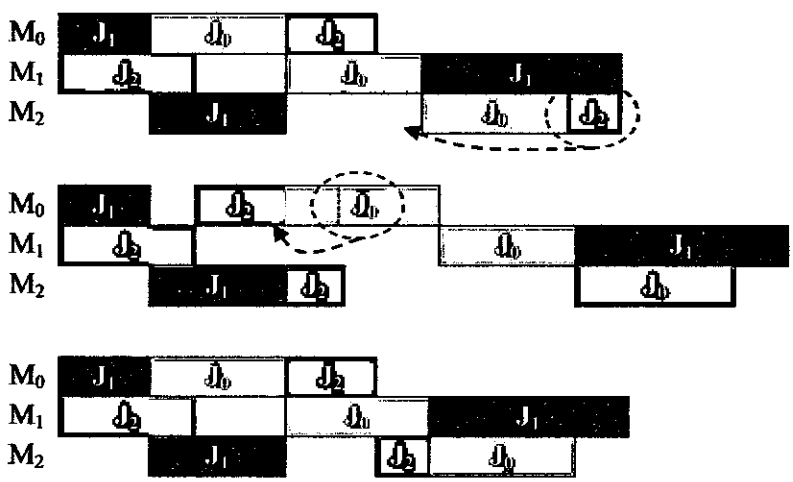


Figura N° 2.3: Planes de de trabajo

En la segunda gráfica se muestra un plan de trabajo activo. En este caso si se cambia la primera operación del trabajo J_0 con la segunda operación del trabajo $J2$, se hace un cambio que afecta a todo el plan de trabajo. Finalmente, en la tercera gráfica se muestra un plan de trabajo no retardado. En este caso, no existe ningún movimiento que haga que no se retrase ninguna operación.

2.2.6 El modelo matemático

Muchos investigadores reconocen que los problemas de scheduling pueden ser resueltos óptimamente utilizando técnicas de programación matemática y uno de los modelos matemáticos más comunes del Job Shop Problem es la programación lineal entera mixta, propuesto por Alan Manne en 1960. Se compone de un conjunto de restricciones lineales y una función objetivo lineal, pero con la restricción adicional que algunas variables de decisión (y_{ipk}) son enteras. Las variables enteras son binarias y se usan para implementar las restricciones disyuntivas, es decir, que ninguna máquina pueda realizar más de una operación y ninguna operación pueda ser realizada en más de una máquina al mismo tiempo. K corresponde a un valor arbitrario grande que debe ser mayor que la suma de los tiempos de procesamiento de todas las operaciones menos el tiempo más pequeño.

El modelo se representa de la siguiente manera:

Sea t_{ik}: tiempo de inicio de cada operación.

J: conjuto de n trabajos a ser procesados.

M: conjunto de m recursos o máquinas.

O_{ik}: operación del trabajo J_i que debe ser procesado en la máquina M_k por un

período ininterrumpido de tiempo T_{ik}.

Minimizar $C_{max}(t_{ik} + T_{ik}) : \forall J_i \in J, \quad M_k \in M$

Sujeto a:

tiempos de inicio $t_{ik} >= 0$ $\{i,p\} \in J$ $\{k,h\} \in M$

restricción de precedencia: $t_{ik} - t_{ih} >= T_{ik}$ si O_{ih} precede a O_{ik}

restricción disyuntiva $t_{pk} - t_{ik} + K(1 - y_{ipk}) >= T_{ik}$ $y_{ipk} = 1$, si O_{ik} precede a O_{pk}

$$t_{ik} - t_{pk} + K(y_{ipk}) >= T_{ik} \qquad y_{ipk} = 0, \text{ en otro}$$

caso

$$donde\ K > (\sum_{i=1}^{n}\sum_{k=1}^{m} T_{ik} - \min(T_{ik})$$

En este modelo existen 2nm + (n(n-1)m)/2 variables, de las cuales n(n-1)m/2 son binarias. El número de restricciones es n(m-1) + n(n-1)m + 2nm. Si *n* tareas son realizadas por *m* máquinas, entonces existe potencialmente $(n!)^m$ secuencias, aunque muchas de ellas serán infactibles debido a las restricciones. De esta manera, un problema con *n*=20 y *m*=10 tiene 7,2651 x 10^{183} posibles soluciones.

2.3. LOS ALGORITMOS GENÉTICOS

Los Algoritmos Genéticos (AGs) son una nueva forma de solucionar los problemas, principalmente los modelos de optimización no lineal. Sobre el particular, García (2008), cuando se refiere a los Algoritmos Genéticos manifiesta que "los organismos vivos poseen una consumada destreza en la resolución de problemas, esta habilidad la obtienen a través de la evolución". En casi todos los organismos, la evolución se produce mediante dos procesos elementales:

- la selección natural, que determina qué miembros de la población sobrevivirán hasta reproducirse, y
- la reproducción, que garantiza la mezcla y recombinación de sus genes entre la descendencia.

El efecto conjunto de estos dos procesos marca el éxito de 3 billones de años de vida sobre el planeta según lo manifiesta García (2008); muchos organismos son capaces, en sucesivas generaciones, de adaptarse a cambios en su forma de vida, e incluso en su estructura, para combatir cualquier agente que hiciera peligrar su supervivencia.

A través de la historia, la naturaleza ha sido una fuente inagotable de inspiración para el desarrollo técnico y científico, el cual, a cambio, ha contribuido a una mejor comprensión de la propia naturaleza. La idea de diseñar algoritmos, para resolver problemas, basándose en los principios de la evolución de los organismos naturales nace independientemente al lado del Pacífico y al lado del Atlántico hace aproximadamente tres décadas. En Estados Unidos, J. Holland (1975) reportado por García (2008), estudia el fenómeno de la adaptación tal y como ocurre en la naturaleza y desarrolla mecanismos para incorporarla en un sistema computarizado.

Según García (2008) "Holland (1975), en su libro 'Adaptación en sistemas naturales y artificiales', propone una clase de algoritmos adaptativos, ideados como una abstracción de la evolución para resolver problemas prácticos y para servir de modelos computacionales de sistemas naturales evolutivos, métodos que posteriormente recibieron el nombre de Algoritmos Genéticos (AGs)".

En la naturaleza, los procesos evolutivos ocurren cuando se satisfacen las siguientes condiciones:

- Una entidad o individuo tiene la habilidad de reproducirse.
- Hay una población de tales individuos que son capaces de reproducirse.
- Existe alguna variedad, diferencia, entre los individuos que se reproducen.
- Algunas diferencias en la habilidad para sobrevivir en el entorno están asociadas a esa variedad.

Esta diversidad se manifiesta en cambios en los cromosomas de los individuos de la población, y se traslada a la variación de la estructura y el comportamiento de los individuos en su entorno. Estos cambios en la estructura y comportamiento se reflejan en el grado de supervivencia, adaptación, y en el nivel de reproducción. Los individuos que mejor se adaptan a su entorno son los que más sobreviven y más se reproducen. En un periodo de tiempo y con muchas generaciones de individuos, la población llega a contener más

individuos cuyos cromosomas se trasladan a estructuras y a comportamientos adecuados a su entorno, sobreviviendo y reproduciéndose en un alto índice, de forma que con el tiempo, la estructura de individuos en la población cambia debido a la selección natural siendo coincidente con lo que manifiesta (Darwin, 1859), reportado por García (2008).

J. Holland fue uno de los pioneros en la aplicación de los principios en los que se basan los procesos de la evolución natural como reglas en el diseño de sistemas artificiales para la resolución de problemas. Según García (2008), "Holland (1975) proporciona un marco general en el que se muestra cómo aplicar los procesos evolutivos a los sistemas artificiales diseñados a partir de los sistemas naturales". Cualquier problema de adaptación puede formularse generalmente en términos genéticos. Una vez formulado en estos términos, tales problemas pueden resolverse mediante AGs.

Paralelamente a los estudios realizados por Holland, la idea de aplicar los principios en los que se basan los procesos de la evolución natural, como reglas en los procedimientos de búsqueda, se investigó con otros enfoques (Fogel y otros 1966; Rechenberg 1973). Actualmente, todas estas aportaciones iniciales han cristalizado en lo que se denomina computación evolutiva según (Bäck y otros 1997; Eiben y Smith 2003), reportados por García (2008); y recoge las distintas vertientes en el diseño de los algoritmos de evolución como herramientas para el análisis de sistemas no lineales complejos por medio de simulación computacional, y basados en los principios de los procesos de evolución natural.

Los AGs según Goldberg (1989b) y Holland (1975) reportados por García (2008) "son procedimientos adaptativos para la búsqueda de soluciones en espacios complejos, inspirados en los procesos genéticos de los organismos naturales y en los principios de la evolución natural de poblaciones". La idea básica es mantener una población de cromosomas, los cuales representan soluciones candidatas a un problema concreto, que evolucionan con el tiempo a través de un proceso de competición y variación controlada. Cada cromosoma tiene una bondad o adaptación asociada, que describe la adecuación de la

solución a la que representa. El proceso de competición, denominado mecanismo de selección, utiliza estas adaptaciones para determinar los cromosomas que se usan para crear otros nuevos. Los nuevos cromosomas se generan a través de los operadores genéticos denominados cruce y mutación.

Los AGs se han aplicado con éxito en problemas de búsqueda y optimización. Gran parte de este éxito se debe a su capacidad para explotar la información acumulada sobre un espacio de búsqueda, y así dirigir las siguientes búsquedas hacia los mejores subespacios. Ésta es su principal ventaja, sobre todo en espacios grandes, complejos y parcialmente definidos donde las técnicas clásicas de búsqueda no son apropiadas. Los AGs ofrecen una aproximación válida a problemas que requieren técnicas de búsquedas efectivas y eficientes.

2.4. QUÉ SON LOS ALGORITMOS GENÉTICOS

Los AG son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proceso genético de los organismos vivos. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza de acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin (1859). Por imitación de este proceso, los AG son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas.

En la naturaleza los individuos de una población compiten entre sí en la búsqueda de recursos tales como comida, agua y refugio. Incluso los miembros de una misma especie compiten a menudo en la búsqueda de un compañero. Aquellos individuos que tienen más éxito en sobrevivir y en atraer compañeros tienen mayor probabilidad de generar un gran número de descendientes. Por el contrario individuos poco dotados producirán un menor número de descendientes. Esto significa que los genes de los individuos mejor

adaptados se propagaran en sucesivas generaciones hacia un número de individuos creciente. La combinación de buenas características provenientes de diferentes ancestros, puede a veces producir descendientes "supe individuos", cuya adaptación es mucho mayor que la de cualquiera de sus ancestros. De esta manera, las especies evolucionan logrando unas características cada vez mejor adaptadas al entorno en el que viven.

El poder de los AG proviene del hecho de que se trata de una técnica robusta, y pueden tratar con éxito una gran variedad de problemas provenientes de diferentes áreas, incluyendo aquellos en los que otros métodos encuentran dificultades. Si bien no se garantiza que el AG encuentre la solución óptima del problema, existe evidencia empírica de que se encuentran soluciones de un nivel aceptable, en un tiempo competitivo con el resto de algoritmos de optimización combinatoria.

En el caso de que existan técnicas especializadas para resolver un determinado problema, lo más probable es que superen al AG, tanto en rapidez como en eficacia. El gran campo de aplicación de los AG se relaciona con aquellos problemas para los cuales no existen técnicas especializadas. Incluso en el caso en que dichas técnicas existan, y funcionen bien, pueden efectuarse mejoras de las mismas hibridándolas con los AG.

Carlos A. Coello Coello, de la Universidad de Tulane, Nueva Orleans, EUA, Manifiesta que un algoritmo genético es un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto.

2.5. ETAPAS DE UN ALGORITMO GENÉTICO

Las etapas de un algoritmo genético son cuatro: Aptitud, Selección, Reproducción, Mutación.

2.5.1 Aptitud

Esta etapa se caracteriza porque gran parte de la inventiva del diseño de un algoritmo descansa en esta etapa, específicamente en cómo se diseña el cromosoma que representará a cada individuo, cromosoma que contendrá de manera implícita las restricciones del problema, para tal efecto se construye también una Función que avalúe a cada individuo, esta es conocida como función de calidad o Fitness.

El diseño del cromosoma requiere de una codificación que es una forma de representar a cada individuo (una posible solución), en esta representación están codificadas todas las características genéticas del problema. Luego se decodifica el cromosoma para encontrar el valor de las características genéticas que se encuentran codificadas en el cromosoma las que finalmente son utilizadas en la Función de Calidad para evaluar a los individuos.

2.5.2 Selección

Esta etapa es encargada de transmitir y conservar aquellas características de las soluciones que se consideran valiosas a lo largo de las generaciones. El principal medio para que la información útil se transmita es que aquellos individuos mejor adaptados (mejor valor de función de evaluación) tengan más probabilidades de reproducirse. Sin embargo, es necesario también incluir un factor aleatorio que permita reproducirse a individuos que aunque no estén muy bien adaptados, puedan contener información útil para posteriores generaciones, con el objeto de mantener así también una cierta diversidad en cada población. Algunas de las técnicas para implementar esta etapa son las siguientes:

Mecanismo *de Selección por Ruleta ó Selección Proporcional.*- Con este método la probabilidad que tiene un individuo de reproducirse es proporcional a su valor de la función de evaluación, es decir, a su adaptación. En este método se define un rango con las características de la

selección por sorteo. El número al azar será un número aleatorio forzosamente menor que el tamaño del rango. El elemento escogido será aquel en cuyo rango esté el número resultante de sumar el número aleatorio con el resultado total que sirvió para escoger el elemento anterior. El comportamiento es similar al de una ruleta, donde se define un avance cada tirada a partir de la posición actual, tiene la ventaja de que no es posible escoger dos veces consecutivas el mismo elemento, y que puede ser forzado a que sea alta la probabilidad de que no sean elementos próximos en la población.

Selección *por Ranking:* Desarrollado por Whitley (1989) consiste en calcular las probabilidades de reproducción atendiendo a la ordenación de la población por el valor de adaptación en vez de atender simplemente a su valor de adecuación. Estas probabilidades se pueden calcular de diversas formas, aunque el método habitual es el ranking lineal (Baker, 1985).

Selección *por Torneo:* Reporta un valor computacional muy bajo debido a su sencillez. Se selecciona un grupo de t individuos (normalmente t = 2, torneo binario) y se genera un número aleatorio entre 0 y 1. Si este número es menor que un cierto umbral K (usualmente 0,75), se selecciona para reproducirse al individuo con mejor adaptación, y si este número es menor que K, se selecciona, por el contrario, al individuo con peor adaptación.

Esta técnica tiene la ventaja de que permite un cierto grado de elitismo -el mejor nunca va a morir, y los mejores tienen más probabilidad de reproducirse y de emigrar que los peores- pero sin producir una convergencia genética prematura, si la población es, al menos, un orden de magnitud superior al del número de elementos involucrados en el torneo. En caso de que la diferencia sea menor no hemos observado mucha diferencia entre emplear el torneo o no.

2.5.3 Reproducción

El operador de cruce permite realizar una exploración de toda la información almacenada hasta el momento en la población y combinarla para crear mejores individuos. Dentro de los métodos habituales destacamos los siguientes:

Cruce de un punto: En la figura N°2.4, se puede observar mediante una flecha llena la selección aleatoria de una posición en los cromosomas es decir en la cadena de los progenitores luego se intercambian los genes a la izquierda de esta posición.

Figura 2.4: Cruza mediante un punto

Cruce de n puntos: Es una generalización del método anterior. Se seleccionan aleatoriamente varias posiciones (n) en las cadenas de los progenitores y se intercambian los genes a ambos lados de estas posiciones. Para n=2, ver Figura 2.5 en las que las posiciones aleatorias se muestran como flechas llenas verticales para cruza mediante dos puntos.

Figura 2.5: Cruza mediante dos puntos

En este caso puede ocurrir que los dos puntos coincidan, de ser así se realiza la cruza como si se tratara de un solo punto.

Cruce Uniforme: Se realiza un test aleatorio para decidir de cuál de los progenitores se toma cada posición de la cadena. El cruce uniforme es una técnica completamente diferente. Cada gen de la descendencia tiene las mismas probabilidades de pertenecer a uno u otro padre.

Aunque se puede implementar de muy diversas formas, la técnica implica la generación aleatoria de una máscara de cruce con valores binarios. Si en una de las posiciones de la máscara hay un uno (1), el gen situado en esa posición en uno de los descendientes se copia del primer padre. Si por el contrario hay un cero (0), el gen se copia del segundo padre. Para producir el segundo descendiente se intercambian los papeles de los padres, o bien se intercambia la interpretación de los unos y los ceros de la máscara de cruce.

Tal y como se puede apreciar en la Figura N° 2.6, la máscara que es generada aleatoriamente contiene 0s y 1s y se encuentra en la parte superior, la descendencia contiene una mezcla de genes de cada uno de los padres. El número efectivo de puntos de cruce es fijo pero será por término medio L/2, siendo L la longitud del cromosoma (número de alelos en representaciones binarias o de genes en otro tipo de representaciones). Se suele referir a este tipo de cruce con las siglas UPC (Uniform Point Crossover).

Máscara	1	0	0	1	0	1	1
Padre 1	A	B	C	D	E	F	G
Padre 2	1	2	3	4	5	6	7
Hijo 1	A	2	3	D	5	F	G
Hijo 2	1	B	C	4	E	6	7

Figura N° 2.6: Cruza Uniforme

Los problemas de optimización combinatoria son muy variados, y no existe un prototipo, pero lo que si tienen en común todos ellos es que el orden de los datos, valores o variables depende directamente del problema o del contexto.

En estos problemas normalmente el espacio de soluciones está formado por ordenaciones de números naturales. La representación más adecuada de los individuos es mediante permutaciones de elementos. Si tenemos que representar K variables, normalmente utilizaremos una lista de K enteros en la que cada valor aparece una única vez; sin embargo la simplicidad de esta representación contrasta con la imposibilidad de utilizar la cruza o reproducción de la manera como se ha definido anteriormente pues porque se generarían individuos no válidos (por ejemplo con valores repetidos). En este caso se utilizan procedimientos de cruza en forma específica para problemas de optimización combinatoria con permutaciones.

Cruce por emparejamiento parcial (PMX): Toma una subcadena del genoma del padre y procura preservar el orden absoluto de los fenotipos - es decir, orden y posición en el genoma- del resto del genoma lo más parecido posible de la madre. Dados dos padres, este operador de cruce copia una subcadena de uno de ellos directamente en las mismas posiciones del otro. Las posiciones restantes se llenan con los valores que aún no hayan sido utilizadas, en el mismo orden que se encontraban en su progenitor. Los pasos a seguir son:

- ✓ Elegir aleatoriamente dos puntos de cruce
- ✓ Intercambiar las dos subcadenas comprendidas entre dichos puntos en los hijos que se generan.
- ✓ Para los valores que faltan en los hijos se copian los valores de los padres
  - Si un valor no está en la subcadena intercambiada se copia igual.
  - Si está en la subcadena intercambiada, entonces se substituye por el valor que tenga dicha subcadena. La Figura N° 2.7, muestra los pasos indicados anteriormente.

Puntos de cruce ↓ ↓

Padre 1	1	3	6	4	2	5	8	7
Padre 2	2	7	8	3	1	5	4	6

Hijo 1				3	1	5		
Hijo 2				4	2	5		

Hijo 1			6	3	1	5	8	7
Hijo 2		7	8	4	2	5		6

Hijo 1	2	4	6	3	1	5	8	7
Hijo 2	1	7	8	4	2	5	3	6

Figura N° 2. 7: Cruza por Emparejamiento Parcial

Cruce de orden: toma una subcadena del genoma del padre 1 y procura preservar el orden relativo de los fenotipos del resto del genoma lo más parecido posible del padre 2. Consiste en copiar en cada uno de los hijos una subcadena de uno de los padres mientras se mantiene el orden relativo que aparece en el otro padre, la secuencia de eventos es como sigue:

- ✓ Elegir aleatoriamente dos puntos de corte.
- ✓ Copiar los valores de las subcadenas comprendidas entre dichos puntos en los hijos que se generan.
- ✓ Para los valores que faltan en los hijos se copian los valores de los padres comenzando a partir de la zona copiada y respetando el orden:
  - Si un valor no está en la subcadena intercambiada, se copia igual.
  - Si está en la subcadena intercambiada, entonces se pasa al siguiente posible.

La Figura N° 2.8, muestra los pasos a seguir para la reproducción por orden.

Puntos de cruce ↓ ↓

Padre 1	1	3	6	4	2	5	8	7
Padre 2	2	7	8	3	1	5	4	6

Hijo 1	8	3	1	4	2	5	6	7
Hijo 2	6	4	2	3	1	5	8	7

Figura N° 2. 8: Cruza de Orden

2.5.4 Mutación

Esta etapa se aplica a un solo individuo a la vez y a toda la población; consiste en cambiar el valor de una de las posiciones de la cadena: si es cero se cambia a uno, y si es uno se cambia a cero la Figura No 2.9, muestra en la posición de la flecha vertical un bit que muta en este caso se muestra cuando cambia de color el bit.

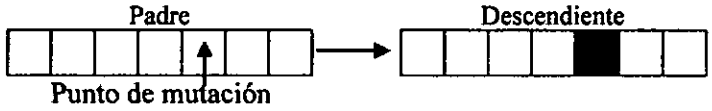


Figura N° 2.9: Mutación de un solo bit

Para cada posición de cada individuo comprobamos la tasa de mutación, y si se cumple cambiamos el bit de cero a uno ó de uno a cero. Habitualmente la tasa de mutación es bastante pequeña en torno al 0.1% comparada con el porcentaje de cruza. Sin embargo esto no es una norma general. En muchos casos la mutación produce individuos con peor adaptación que los individuos originales, ya que la mutación puede romper las posibles correlaciones entre genes que se hayan formado con la evolución de la población. Sin embargo, contribuyen a mantener la diversidad de la población, que es fundamental para el buen funcionamiento del algoritmo.

Cuando la representación de los individuos no es binaria, la mutación consiste en el intercambio de genes, en un mismo individuo en la Figura N° 2.10 se puede observar la posición de dos genes indicados con la flecha vertical y además el color de cada gene es diferente y luego del intercambio (mutación), se observa el descendiente que a cambiado los colores.

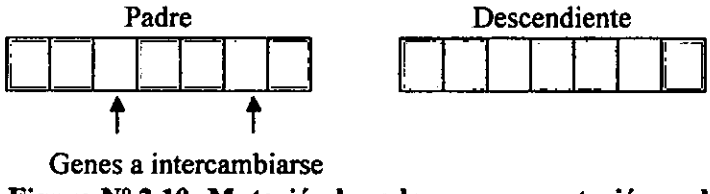


Figura N° 2.10: Mutación basada en representación no binaria

CAPÍTULO 3: METODOLOGÍA DE LA INVESTIGACIÓN

3.1 TIPO DE INVESTIGACIÓN

El método científico en cualquier investigación exige claramente definir el tipo de investigación, en este trabajo se definió el enfoque **cuantitativo** en razón a que se usó la recolección de diferentes prototipos de problemas para evaluar los resultados de la aplicación del algoritmo genético construido, medidos en función del tiempo de respuesta y la calidad de las soluciones; observándose los parámetros del algoritmo que permitieron determinar los valores óptimos. Así mismo el proceso de la investigación fue **deductivo** en razón a que se utilizó una metodología de propósito general como es la Teoría de la Evolución de los seres vivos de Darwin aplicada a un problema particular como es el Job Shop Scheduling Problem (JSSP).

3.2 DISEÑO DE LA INVESTIGACIÓN

El diseño de la investigación fue de tipo **experimental** porqué se probó el algoritmo utilizando un diseño factorial para calibrar los parámetros óptimos del mismo y se resolvió el problema de planificación detallada de operaciones del tipo Job Shop. En estas condiciones se puede afirmar que la investigación también fue de tipo explicativa para lo cual se definieron las siguientes etapas:

a) En primer lugar se realizó una investigación documental del entorno en que se desarrolla el problema de planificación detallada de operaciones del tipo Job Shop definiendo claramente sus parámetros y

b) En segundo lugar se establecieron 30 instancias como ejemplos prototipos de aplicación a resolver.

c) Así mismo se diseñó la propuesta de solución mediante el algoritmo genético definiendo claramente el diseño del cromosoma y la función de calidad que

evaluó a cada individuo dentro de la población. Así mismo se desarrolló las funciones necesarias a implementarse en el algoritmo genético computacional, encontrándose respuestas altamente satisfactorias para los casos de estudio y se efectuó un análisis comparativo desde el punto de vista de eficiencia, en términos de encontrar la solución, tiempo de respuesta y tamaño del problema a resolver.

3.3 MATERIALES Y MÉTODOS

MATERIALES

Para cumplir este propósito se utilizarán los siguientes equipos y material:

1) Una computadora.
2) Impresora.
3) Software Visual Basic 6.0 como lenguaje de programación.
4) Software disponible como MS Excel entre otros
5) Bibliografía especializada en Investigación de Operaciones.
6) Bibliografía especializada sobre Algoritmos Genéticos.

DISEÑO PARA LA PRUEBA DE HIPÓTESIS

Para probar la eficiencia de los Algoritmos Genéticos, se utilizará la técnica del análisis de varianza de un diseño factorial de a x b tratamientos con 4 repeticiones. El número de tratamientos en estudio estará representado por dos factores y cuatro niveles cada uno como son:

a) Tamaño de la Población en el Algoritmo Genético.
b) Número Generaciones hasta la convergencia.

La Variable de respuesta en el diseño experimental es el cálculo del makespan (tiempo mínimo de ejecutar todos los trabajos) y adicionalmente se calcula el tiempo que le toma al algoritmo encontrar el makespan.

UNIDAD EXPERIMENTAL

La unidad experimental estará constituida por cada ejemplo prototipo o instancia del problema, que se desea resolver, para garantizar la estabilidad del algoritmo genético.

ANÁLISIS DE VARIANZA

Fuentes de variación		**G.L**
Tratamientos		**15**
A	: Factor Tamaño de la población (a-1)	3
B	: Factor Número de generaciones (b-1)	3
AxB	: Interacción (a-1)x(b-1)	9
Error experimental		**48**
Total	(abc-1)	**63**

3.4 FASES DEL DESARROLLO DE LA INVESTIGACIÓN

3.4.1 Definición de instancias de problemas de prueba

Para la calibración del algoritmo se inició con instancias de prueba generadas aleatoriamente, luego se utilizó instancias de prueba reportados en diferentes papers para verificar la eficacia del algoritmo para tal efecto y con el propósito de determinar los parámetros más efectivos del comportamiento del algoritmo se utilizó el diseño experimental que se planteó anteriormente.

Las instancias de prueba están referidas a 11 ejemplos de índole académico, como prototipos de problemas y que se muestran a continuación y cuyos resultados del algoritmo junto al análisis de varianza se muestra en el anexo N° 2.

Las instancias de prueba muestran la información de cada prototipo en filas y columnas en el que, cada columna corresponde a un trabajo y

cada fila a una máquina; el número entre una fila y una columna representa el tiempo de operación del trabajo en la máquina; en ambos casos los trabajos y las máquinas comienzan con el valor 1, es decir trabajo 1 (T1), ó máquina 1 (M1). En la parte inferior de cada instancia se muestra la secuencia de operación de los trabajos en las máquinas comenzando con el trabajo 1 a la derecha del trabajo aparece la secuencia de máquinas por donde debe pasar cada trabajo, y así sucesivamente:

INSTANCIA 1

	T1	T2	T3	T4
M1	08	05	06	07
M2	02	04	01	03
M3	04	05	03	02

S1:	M1	M2	M3
S2:	M1	M2	M3
S3:	M1	M2	M3
S4:	M1	M2	M3

INSTANCIA 2

	T1	T2	T3	T4	T5
M1	10	05	02	06	06
M2	05	08	10	09	07

S1:	M2	M1
S2:	M1	M2
S3:	M1	M2
S4:	M2	M1
S5:	M1	M2

INSTANCIA 3

	T1	T2	T3	T4	T5
M1	07	08	05	04	10
M2	07	08	02	02	03
M3	06	04	06	10	08

S1:	M3	M1	M2
S2:	M2	M1	M3
S3:	M3	M2	M1
S4:	M1	M2	M3
S5:	M1	M2	M3

INSTANCIA 4

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
M1	07	10	07	08	04	06	05	05	09	00
M2	05	08	05	09	09	05	03	04	04	09
M3	05	09	07	10	05	08	05	08	04	00
M4	06	09	05	08	08	04	07	10	01	02
M5	02	03	01	07	03	01	01	00	08	00

S1:	M4	M5	M2	M3	M1
S2:	M1	M3	M4	M2	M5
S3:	M5	M1	M2	M3	M4
S4:	M5	M2	M4	M3	M1
S5:	M3	M2	M4	M5	M1
S6:	M2	M4	M5	M3	M1
S7:	M4	M5	M1	M2	M3
S8:	M1	M2	M4	M3	M5
S9:	M1	M2	M3	M4	M5
S10:	M3	M2	M1	M4	M5

INSTANCIA 5

	T1	T2	T3	T4	T5
M1	10	06	09	09	07
M2	09	07	08	05	04
M3	04	08	09	00	07
M4	01	01	01	03	00
M5	08	03	03	09	04
M6	04	07	00	03	04
M7	04	02	02	09	06
M8	08	02	09	05	10
M9	06	05	08	03	01
M10	00	03	02	07	07

S1:	M7	M8	M4	M1	M5	M9	M10	M2	M3	M6
S2:	M7	M9	M10	M8	M2	M5	M1	M6	M3	M4
S3:	M2	M6	M5	M1	M7	M8	M3	M9	M4	M10
S4:	M6	M5	M9	M7	M4	M8	M3	M10	M1	M2
S5:	M1	M6	M3	M9	M7	M2	M4	M5	M10	M8

INSTANCIA 6

	T1	T2	T3	T4	T5	T 6	T 7
M1	10	02	00	06	03	04	07
M2	05	00	01	08	04	05	03
M3	01	02	09	05	10	10	08
M4	07	00	06	07	07	05	03
M5	09	07	08	03	09	06	06
S0	M	M	M	M	M		

INSTANCIA 7

	T1	T2	T3	T4	T5		
M1	09	02	03	08	08		
M2	02	01	08	02	09		
M3	04	00	10	03	01		
M4	09	06	02	03	09		
M5	09	04	04	07	07		
M6	10	04	09	06	01		
M7	07	04	05	07	06		

I	5	4	2	3	1		
S0 2	M 5	M 3	M 4	M 1	M 2		
S0 3	M 4	M 2	M 3	M 5	M 1		
S0 4	M 4	M 5	M 1	M 2	M 3		
S0 5	M 3	M 1	M 2	M 4	M 5		
S0 6	M 4	M 3	M 2	M 5	M 1		
S0 7	M 5	M 2	M 4	M 3	M 1		

S0 1	M 7	M 4	M 6	M 5	M 1	M 2	M 3
S0 2	M 5	M 6	M 7	M 2	M 1	M 3	M 4
S0 3	M 4	M 6	M 3	M 5	M 7	M 1	M 2
S0 4	M 1	M 2	M 5	M 3	M 4	M 6	M 7
S0 5	M 4	M 5	M 3	M 2	M 6	M 1	M 7

INSTANCIA 8						
	T1	T2	T3	T4	T5	T6
M1	01	08	05	02	03	02
M2	03	05	04	01	02	04
M3	06	10	08	03	05	06
M4	07	10	09	04	06	01
M5	03	10	01	05	01	05
M6	06	04	07	06	04	03
S0 1	M 3	M 1	M 2	M 4	M 6	M 5
S0 2	M 2	M 3	M 5	M 6	M 1	M 4
S0 3	M 3	M 4	M 6	M 1	M 2	M 5
S0 4	M 2	M 1	M 3	M 4	M 5	M 6
S0 5	M 3	M 2	M 5	M 6	M 1	M 4
S0 6	M 2	M 4	M 6	M 1	M 5	M 3

INSTANCIA 9								
	T1	T2	T3	T4	T5	T6	T 7	T 8
M1	03	02	01	02	05	04	00	01
M2	01	02	03	03	03	02	04	00
M3	03	03	06	02	01	01	02	00
M4	02	04	02	03	05	01	00	01
M5	04	02	05	03	01	02	01	00
M6	03	01	01	03	03	03	02	02
S0 1	M 1	M 6	M 2	M 4	M 3	M 5		
S0 2	M 3	M 4	M 2	M 6	M 5	M 1		
S0 3	M 5	M 3	M 2	M 6	M 4	M 1		
S0 4	M 1	M 2	M 5	M 6	M 4	M 3		
S0 5	M 4	M 2	M 5	M 1	M 3	M 6		
S0 6	M 2	M 6	M 1	M 5	M 4	M 3		
S0 7	M 3	M 5	M 6	M 2	M 1	M 4		
S0 8	M 6	M 1	M 4	M 3	M 2	M 5		

INSTANCIA 10

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13
M1	09	00	00	00	07	00	00	00	00	00	00	00	00
M2	00	07	00	00	00	09	00	00	00	00	00	00	00
M3	00	00	00	00	00	00	00	00	09	04	01	00	02
M4	00	00	06	01	00	00	00	06	00	00	03	00	00
M5	00	05	00	03	00	00	08	00	00	00	00	00	00
M6	07	00	00	00	09	00	00	00	00	00	00	00	00
M7	00	00	00	04	00	07	05	00	00	00	00	00	00
M8	00	00	00	03	00	00	02	00	00	09	02	00	00
M9	00	00	06	00	00	00	00	10	00	00	00	00	00
M10	00	04	00	00	00	00	00	00	05	00	02	05	00

S01	M1	M6		
S02	M2	M10	M5	
S03	M4	M9		
S04	M5	M4	M7	M8
S05	M6	M1		
S06	M7	M2		
S07	M8	M5	M7	
S08	M9	M4		
S09	M10	M3		
S10	M3	M8		
S11	M8	M3	M10	M4
S12	M10			
S13	M3			

INSTANCIA 11

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
M1	03	01	11	00	03	09	00	08	13	02
M2	08	10	13	01	00	08	06	10	06	06

S01	M2	M1
S02	M1	M2
S03	M2	M1
S04	M2	M1
S05	M1	M2
S06	M1	M2
S07	M2	M1
S08	M2	M1
S09	M2	M1
S10	M1	M2

3.4.2 Definición de los parámetros de entrada para el algoritmo

La definición de los parámetros para el algoritmo depende del problema que va resolver, sin embargo en todos los casos se encontraron parámetros comunes que han sido considerados tales como:

- Datos de entrada relativos al problema
- Datos de entrada relativos al algoritmo
- Datos de salida que el algoritmo debe reportar

Datos de entrada relativos al problema.- Dependen estrictamente del problema

Datos de entrada relativos al algoritmo

Tamaño de la Población.- Es un número entero múltiplo de 2 y permite inicializar la población de posibles soluciones; puede variar entre 10 y 200.

Número de Generaciones.- Número entero positivo que puede variar de 10 a 200, permite encontrar la mejor solución dentro del intervalo.

Porcentaje de Mutación.- Operador Genético que es igual o menor al 10%

Porcentaje de Cruza.- Operador Genético que se encuentra en el intervalo 80% a 95%.

Números de puntos de cruce.- Se generan dos puntos enteros positivos que pueden tomar los valores desde 1 hasta tamaño del cromosoma menos 1.

Datos de Salida

Los datos de salida están relacionados con el resultado de aplicar el algoritmo genético y está en términos de encontrar la solución óptima y también depende del problema, en todo caso se reportará la

mejor solución en el que está representado un diagrama de Gantt y el makespan.

3.4.3 Diseño del cromosoma y función de calidad

Antes de presentar el diseño del cromosoma se presenta un prototipo del problema que se va a resolver, por ejemplo si se tiene 5 trabajos y 3 máquinas que tienen que procesarse de acuerdo a los tiempos de operación en cada una de las máquinas y a la secuencia predeterminada, el cuadro Nº 3.1, ilustra este ejemplo:

Cuadro Nº 3.1: Ejemplo prototipo del JSSP a resolver

Tiempos de operación (min), de trabajos en las máquinas y secuencia a

		TRABAJOS				
		0	1	2	3	4
MÁQUINAS	M	0	0	0	0	1
	M	0	0	0	0	0
	M	0	0	0	1	0

	SECUENCIA		
T0	M	M	M
T1	M	M	M
T2	M	M	M
T3	M	M	M
T4	M	M	M

El diseño del cromosoma que contenga una solución al problema JSSP, se construyó partiendo de la representación del problema del agente viajero (PAV), en el que el cromosoma está compuesto por una cadena de números que representa las ciudades a visitar las cuales no pueden repetirse; en buena cuenta es una permutación sin repetición. En el caso particular del JSSP, el cromosoma está compuesto por un arreglo matricial de doble entrada en el que las filas representan las máquinas y las columnas los trabajos por tanto la dimensión del cromosoma será una matriz de orden m x n, en donde m representa el número de máquinas y n el número de trabajos.

Cómo puede observarse en la figura N° 3.1, el cromosoma está compuesto por 3 máquinas y 5 trabajos, en este caso cada máquina m_i, contiene una permutación de trabajos J_j sin repetición que tiene que trabajarse en la máquina.

	PERMUTACIÓN SIN REPETICIÓN DE LOS TRABAJOS EN LAS				
M0	0	3	2	4	1
M1	3	0	1	4	2
M2	4	3	2	0	1

Figura N° 3.1: Cromosoma

Para construir el diagrama de Gantt a partir de esta representación se decodifica el cromosoma formando una sola cadena comenzando primero por la primera columna de los trabajos y se ubican manteniendo el orden esto es: la primera columna de los trabajos es 0 3 4, lo que se pone primero luego se sigue con la segunda columna esto es 3 0 3, y así sucesivamente hasta terminar con la última columna quedando la cadena que se ve la Tabla N° 3.1.

Tabla N° 3.1: Decodificación del cromosoma

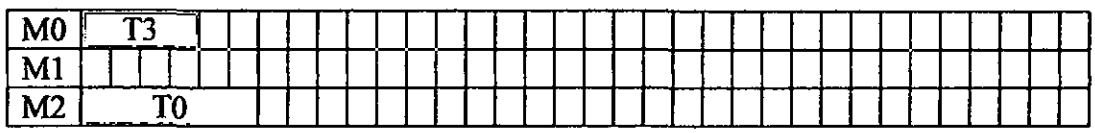
Columna 1			Columna 2			Columna 3			Columna 4			Columna 5		
0	3	4	3	0	3	2	1	2	4	4	0	1	2	1

Esta cadena contiene el diagrama de Gantt y se construye de la manera siguiente:

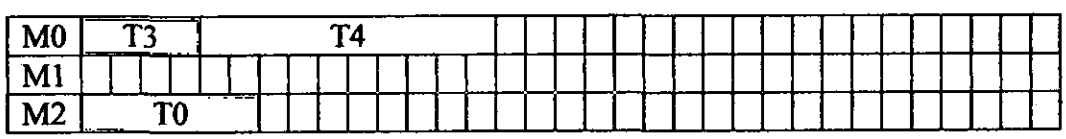
1°) Primero se debe procesar el trabajo N° 0 para ello se busca en la secuencia (es información que se ingresa con el problema) la máquina con la que se inicia el trabajo 0, esto es máquina 2, y se ubica en el diagrama de Gantt de acuerdo al tiempo que demanda esta operación así:

M0																														
M1																														
M2	T0																													

2°) Como el segundo número en la cadena es 3, se busca en la secuencia del trabajo 3 con que máquina inicia observándose que se inicia con la máquina 0, luego se grafica de acuerdo al tiempo requerido.



3°) Luego se continúa con el trabajo N° 4 (3er número en la cadena), y la secuencia de este trabajo indica que se inicia en la máquina 0, entonces se continua en la máquina 0 inmediatamente, y se grafica de acuerdo al tiempo requerido.



Y así sucesivamente hasta construir el diagrama de Gantt que se muestra a continuación en la Figura N° 3.4, observándose que todos los trabajos son terminados con un makespan es de 34 minutos.

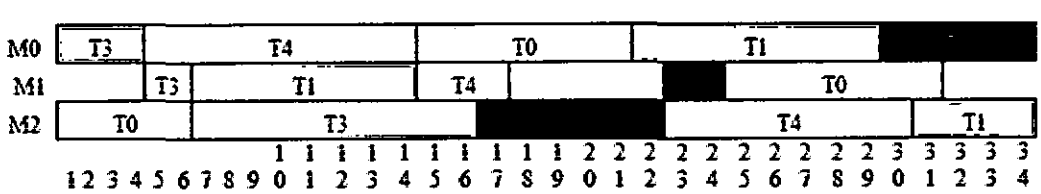


Figura N° 3.4: Decodificación del cromosoma y su makespan

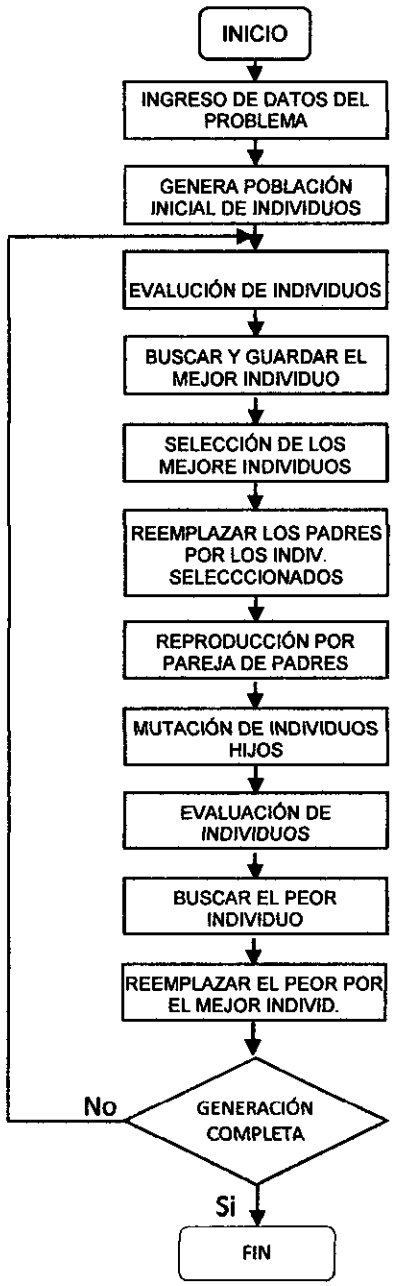
LA FUNCIÓN DE CALIDAD FITNESS

La función de calidad que evalúa a cada individuo de la población se apoya en el cálculo del makespan para ello se determina primero el tiempo máximo de realizar todas las operaciones considerando que cada operación continúa una después de la otra y sin traslape, lo cual es equivalente a colocar todas las operaciones en una misma línea y el tiempo máximo se calcula sumando todos los tiempos de todas las operaciones en todas las máquinas; por consiguiente:

Fitness = tiempo máximo - makespan

Cómo tiempo máximo = 90, Fitness = 90 – 34 = 56

3.4.4 Diagrama de flujo del Algoritmo



Este diagrama de flujo se utiliza para desarrollar el código y su implementación computacional del algoritmo genético simple.

3.4.5 Método de reproducción utilizado en el algoritmo genético

En el proceso de reproducción, se utilizó la técnica de cruza por emparejamiento parcial en razón a que esta técnica mantiene los individuos hijos válidos sin necesidad de reconstrucción, además porque cada individuo tiene una estructura genética formada por subcadenas que son resultado de una permutación sin repetición. Por consiguiente en el proceso de reproducción se crean dos puntos de cruce por cada máquina y estos puntos pueden ser diferentes de una máquina a la siguiente lográndose una alta diversidad de búsqueda de nuevos individuos, a partir de la trasmisión genética de los padres, para mayor claridad se muestra cómo se realiza la reproducción a partir de dos individuos partiendo del ejemplo prototipo del cuadro N° 3.1.

M0	J4	J3	J2	J0	J1
M1	J2	J0	J3	J4	J1
M2	J3	J0	J4	J1	J2

Padre1

M0	J2	J1	J3	J0	J4
M1	J3	J2	J1	J4	J0
M2	J4	J3	J0	J2	J1

Padre 2

Se generan 3 pares de números aleatorios entre 0 y 4; por ejemplo: (1,3), (2,4) y (1,4)

Reproducción de máquina cero: M0 en los puntos (1,3)

P1: J4 J3 J2 J0 J1 H1: J1 J3 J0 J4 J1 J3 J0 J2

P2: J2 J1 J3 J0 J4 H2: J3 J2 J0 J1 J3 J2 J0 J4

Reproducción de máquina uno: M1 en los puntos (2,4)

P1: J2 J0 J3 J4 J1 H1: J1 J4 J0 J2 J3 J1 J4 J0

P2: J3 J2 J1 J4 J0 H2: J3 J4 J1 J0 J2 J3 J4 J1

Reproducción de máquina dos: M2 en los puntos (1,4)

P1: J3 J0 J4 J1 J2 H1: J3 J0 J2 J1 J4 J3 J0 J2

J1

P2: J4 J3 J0 J2 J1 H2: J0 J4 J1 J2 J3 J0 J4 J1

J2

M0	J4	J1	J3	J0	J2
M1	J2	J3	J1	J4	J0
M2	J4	J3	J0	J2	J1

Hijo1

M0	J2	J1	J3	J0	J4
M1	J3	J2	J1	J4	J0
M2	J3	J0	J4	J1	J2

Hijo 2

Estos Hijos luego de ser decodificados siguen el proceso de mutación y luego se les evalúa para calcular el Fitness y el makespan.

3.4.6 Mutación de individuos

Para efectuar la mutación en un individuo se genera dos números aleatorios por máquina, luego se intercambian dentro de cada máquina los genes correspondientes a los números aleatorios, se repite este proceso tantas veces como máquinas existen en el individuo; de esta manera se mantiene la integridad del individuo después de mutarse, tal como se puede observar en la Figura N° 3.5, en el que se ha generado tres pares de números aleatorios; para mutar el individuo generado anteriormente como Hijo1. Debe entenderse que esta etapa solo ocurre al 5% de los individuos o menos de una población.

M0	J4	J1	J3	J0	J2	Números aleatorios
M1	J2	J3	J1	J4	J0	Números aleatorios
M2	J4	J3	J0	J2	J1	Números aleatorios

Hijo1

M0	J4	J0	J3	J1	J2
M1	J2	J3	J0	J4	J1
M2	J4	J0	J3	J2	J1

Hijo1 después de mutarse

Figura N° 3.5: Mutación de Individuo Hijo1

3.4.7 Desarrolló del código

El algoritmo se inicia Ingresando el problema o generándolo aleatoriamente, de cualquier manera que se tenga en memoria un problema a resolverse éste se puede guardarse para ser leído posteriormente. El código que soporta esta propiedad del algoritmo no se comenta porque no es la razón principal del algoritmo, en su lugar se comentará los procedimientos y funciones principales del algoritmo y en el orden de ejecución. El código de todo el algoritmo se encuentra en el anexo N°1.

Generación de individuos.- Utiliza el procedimiento:
"GeneraPoblacionInicial()"

Se inicia el procedimiento ingresando el tamaño de la población, luego se redimensiona los tipos de datos ecosistema y padres de acuerdo al tamaño de la población, se redefine el tamaño de un arreglo para guardar el mejor individuo de cada generación y se generan todos los individuos aleatoriamente. Una vez concluido la creación de la población; cada cromosoma, se decodifica y se guarda en un arreglo unidimensional y habrán tantos arreglos como el tamaño de la población ingresada por el usuario.

Evaluación de la calidad de cada individuo.- Utiliza dos procedimientos:
"EvaluaOperacion()" y "calculaFitness()"

El primer procedimiento lo que hace es decodificar la cadena de cada individuo de la población y calcula el makespan. El segundo procedimiento determina la calidad de cada individuo restando el makespan de cada individuo del tiempo máximo que podría darse para completar todos los trabajos lo cual se consigue sumando los tiempos de todos los trabajos en todas las máquinas. A partir de aquí se entra en un bucle que termina cuando se ha concluido con todas las generaciones o iteraciones indicadas al algoritmo.

Buscar el mejor individuo de la población.- Utiliza el procedimiento: "mejor()"

Este procedimiento busca en toda la población el individuo que tiene el mejor Fitness y lo guarda en un arreglo, así mismo guarda su makespan y su Fitness.

Selección de los mejores individuos.- Utiliza el procedimiento: "Ruleta()"

Procedimiento mediante el cual se ha implementado el mecanismo de selección por ruleta lográndose seleccionar a los individuos que tienen mejor Fitness o mejor adaptación al medio esto origina que algunos individuos se pueden repetir varias veces por tener buena calidad, finalmente en el proceso de selección de los individuos seleccionados en un número igual al tamaño de la población constituyen la población de los padres que se cruzarán posteriormente.

Reproducción o cruza de los individuos seleccionados.- Procedimiento "reproduccion()"

Este procedimiento recorre la mitad de la población porque utiliza parejas de individuos para realizar la cruza utilizando el método expuesto en el ítem 3.4.5, sin embargo no todos las parejas se cruzan es probable que algunas parejas pasan directamente a la siguiente generación sin tener descendencia, en razón a que la probabilidad de cruzamiento del 95%, es la mostró mayor eficiencia en la solución de todas las instancias de prueba.

Mutación de la descendencia.- procedimiento "Mutacion()"

Luego del cruzamiento de los individuos padres, cada uno de los descendientes pasa a la etapa de mutación la cual consiste en la alteración de la estructura de algunos genes del cromosoma del individuo sin embargo como ocurre en la naturaleza estas mutaciones se presentan con muy baja probabilidad en este trabajo se ha determinado que el porcentaje de mutación que mejor comportamiento ha tenido en la solución es de 5%, lo que significa primero generar un número aleatorio entre 0 y 1 y si este

es menor o igual a 0.05 el individuo se mutará. El procedimiento de mutación sigue las indicaciones expuestas en el ítem 3.4.6, la misma que garantiza la integridad del individuo después de mutarse y en consecuencia la integridad de la solución. Luego de la mutación se corren los procedimientos "EvaluaOperacion()" y "calculaFitness()" ; ya comentados.

Mantenimiento de elitismo en el algoritmo.- Procedimiento "peOr()" y "RemplazaPxM()"

Finalmente y con el propósito de mantener elitismo y un mejor comportamiento del algoritmo en los descendientes obtenidos y después de evaluarlos y determinar su calidad se busca el peor individuo y se le reemplaza por el mejor individuos de la generación anterior.

A partir de aquí el proceso es iterativo reiniciándose en "Buscar el mejor individuo de la población" y se repetirá tantas veces conforme al número de generaciones que se haya definido y para este trabajo el número de generaciones que mostró un buen comportamiento es de 80 generaciones.

CAPÍTULO 4: ANÁLISIS Y DISCUCIÓN DE RESULTADOS

4.1 CALIBRACIÓN DEL ALGORITMO

La calibración del algoritmo se realizó generando automáticamente instancias en forma aleatoria entre 3 y 20 trabajos y entre 10 y 15 máquinas con tiempos de ejecución aleatorios entre 0 y 10 unidades de tiempo en el que cada unidad de tiempo puede representar: minutos, cantidades de 10 minutos, cantidades de 30 minutos, horas, etc.; posteriormente se corrió el algoritmo combinando diferentes operadores genéticos como porcentaje de cruza, porcentaje de mutación número de generaciones y tamaño de la población para determinar los operadores genéticos que mejor comportamiento tenían en la solución observándose que los operadores genéticos que mejor se comportaron fueron: porcentaje de cruza 95%, porcentaje de mutación 5%, tamaño de la población de 30 a 60 individuos con saltos de 10, así mismo el número de generaciones entre 50 y 80 con saltos de 10. Efectuada esta calibración se procedió a aplicar el algoritmo a los problemas seleccionados y que se encuentran en los anexos N° 2 y N° 3

4.2 APLICACIÓN DEL ALGORITMO A LAS INSTANCIAS DE PRUEBA

Los resultados de la aplicación del algoritmo a las instancias de prueba se muestran en el anexo N° 2 y Anexo N° 3, sin embargo aquí se muestran y comentan los resultados de 11 instancias a través del diagrama de Gantt y en algunos casos observando el comportamiento de los resultados del algoritmo durante su ejecución de manera gráfica; sobre el particular la presencia de un cero (0), en el diagrama de Gantt, es un indicativo de que no se realiza ningún trabajo es decir la máquina está inactiva, los demás valores indican el número de trabajo que se está procesando en la máquina correspondiente.

En todas las instancias el diagrama de Gantt, muestra al lado izquierdo las máquinas y a la derecha de cada una, números que indican los trabajos a realizarse y la cantidad de veces que se repite indica el tiempo total de operación; cuando el número es cero indica que la máquina está ociosa.

Instancia N°1: 4x3 (4 trabajos 3 máquinas)

Todas las respuestas proporcionadas por el algoritmo son coincidentes con los resultados publicados[1] en la web, consultada el 13/10/2014. Sobre el particular el coeficiente de variabilidad es cero (0), para todas las corridas. Y la solución óptima presenta un makespan de 30, conforme se puede observar en la salida que proporciona el algoritmo como un diagrama de Gantt que se ilustra ver figura N° 4.1.

M01: 02 02 02 02 02 01 01 01 01 01 01 01 04 04 04 04 04 04 03 03 03 03 03 03 00 00 00 00
M02: 00 00 00 00 00 02 02 02 02 00 00 00 00 01 01 00 00 00 00 00 04 04 04 00 00 00 03 00 00 00
M03: 00 00 00 00 00 00 00 00 00 02 02 02 02 02 00 01 01 01 01 00 00 00 00 04 04 00 00 03 03 03
Makespan = 30

Figura N° 4.1: Solución Instancia N°1

Instancia N° 2: 5x2 (5 trabajos 2 máquinas)

Esta instancia es generada aleatoriamente por el algoritmo encontrándose un makespan de 39, el cual se mantiene en todas las corridas del algoritmo originando un coeficiente de variabilidad de 0, para todas las corridas, además se puede observar que existen múltiples soluciones, pero solo se muestran dos con su correspondiente diagrama de Gantt, Figura N° 4.2

M01: 02 02 02 02 02 05 05 05 05 05 05 03 03 00 01 01 01 01 01 01 01 01 01 04 04 04 04 04 04 00 00 00 00 00 00 00 00 00 00 00
M02: 04 04 04 04 04 04 04 04 04 01 01 01 01 01 02 02 02 02 02 02 02 02 05 05 05 05 05 05 05 03 03 03 03 03 03 03 03 03 03 03
Makespan = 39

M01: 05 05 05 05 05 05 05 02 02 02 02 02 03 03 00 04 04 04 04 04 04 01 01 01 01 01 01 01 01 01 01 01 00 00 00 00 00 00 00 00 00 00
M02: 01 01 01 01 01 01 04 04 04 04 04 04 04 04 04 05 05 05 05 05 05 05 05 02 02 02 02 02 02 02 02 03 03 03 03 03 03 03 03 03 03 03
Makespan = 39

Figura N° 4.2: Soluciones Instancia N°2

1 http://www.uelbosque.edu.co/sites/default/files/publicaciones/revistas/revista_tecnologia/volumen3_numero2/algoritmo_genetico3-2.pdf

Instancia N° 3: 5x3 (5 trabajos 3 máquinas)

Instancia generada aleatoriamente, el mínimo makespan es de 34 y se repite en todas las corridas lo que presenta un coeficiente de variabilidad de 0, además también se presenta múltiples soluciones alternativas, conforme se puede apreciar en los diagramas de Gantt que se muestra a continuación:

M01: 04 04 04 04 05 05 05 05 05 05 05 05 05 05 01 01 01 01 01 01 01 02 02 02 02 02 02 02 02 03 03 03 03 03
M02: 00 00 00 00 04 04 02 02 02 02 02 02 02 02 05 05 05 00 00 00 00 00 03 03 01 01 01 01 01 01 01 00 00 00
M03: 01 01 01 01 01 01 04 04 04 04 04 04 04 04 04 04 03 03 03 03 03 03 05 05 05 05 05 05 05 05 02 02 02 02
Makespan = 34

M01: 04 04 04 04 05 05 05 05 05 05 05 05 05 05 01 01 01 01 01 01 01 02 02 02 02 02 02 02 02 03 03 03 03 03
M02: 00 00 00 00 04 04 02 02 02 02 02 02 02 02 05 05 05 03 03 00 00 01 01 01 01 01 01 01 00 00 00 00 00 00
M03: 01 01 01 01 01 01 03 03 03 03 03 03 04 04 04 04 04 04 04 04 04 04 05 05 05 05 05 05 05 05 02 02 02 02
Makespan = 34

Figura N° 4.1: Soluciones Instancia N°3

Instancia N° 4: 10x5 (10 trabajos 5 máquinas)

En esta instancia después de 43 generaciones se alcanza el valor mínimo de makespan llegando a 66 y se mantiene en ese estado hasta terminar con todas las generaciones; sin embargo en este caso las diferentes corridas han arribado a tres valores diferentes en el Makespan a saber: 66, 67 y 68 lo que resulta en un coeficiente de variabilidad respecto de la media del orden de 1.19% sin embargo los diferentes factores en estudio no tienen estadísticamente significación en el Makespan. Conforme se puede apreciar en el análisis de varianza del anexo N° 2. Los diagramas de Gantt con sus respectivos makespan de 66 y 67 se muestran a continuación en las figuras N° 4.2 y N° 4.3 y el comportamiento del algoritmo en las graficas correspondientes de las figuras N° 4.4 y N° 4.5.

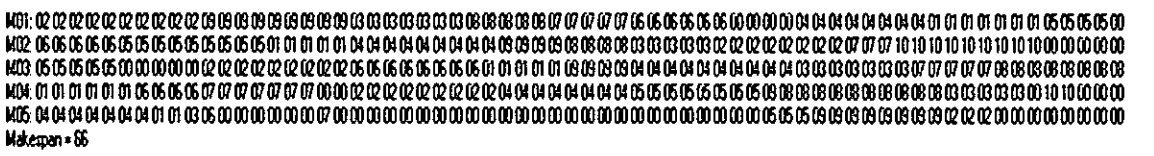


Figura N° 4.2: Soluciones Instancia N°4, con makespan de 66

M01: 02 02 02 02 02 02 02 02 02 02 08 08 08 00 00 09 09 09 09 09 09 09 09 09 03 03 03 03 03 03 03 06 06 06 06 06 06 07 07 07 07 07 00 00 00 01 01 01 01 01 01 01 05 05 05 05 04 04 04 04 04 04 04 04 00 00 00
M02: 06 06 06 06 06 05 05 05 05 05 05 05 05 05 00 08 08 08 08 01 01 01 01 01 09 09 09 09 04 04 04 04 04 04 04 04 04 03 03 03 03 03 10 10 10 10 10 10 10 10 10 07 07 07 02 02 02 02 02 02 02 02 00 00 00 00 00
M03: 05 05 05 05 05 00 00 00 00 00 02 02 02 02 02 02 02 02 02 06 06 06 06 06 06 06 06 00 09 09 09 08 08 00 00 08 08 08 00 08 01 01 01 01 01 04 04 04 04 04 04 04 04 04 04 03 03 03 03 03 03 03 07 07 07 07 07
M04: 07 07 07 07 07 07 07 01 01 01 01 01 01 06 06 06 06 00 00 08 00 03 03 03 08 08 08 00 03 05 05 05 05 05 05 05 05 04 04 04 04 04 04 04 04 02 02 02 02 02 02 02 02 02 08 10 10 00 00 00 00 00 03 03 03 03 03
M05: 03 04 04 04 04 04 04 04 07 00 00 00 00 01 01 00 00 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 05 05 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 09 09 09 09 09 09 09 09 02 02 02 00
Makespan = 67

Figura N° 4.3: Soluciones Instancia N°4, con makespan de 67

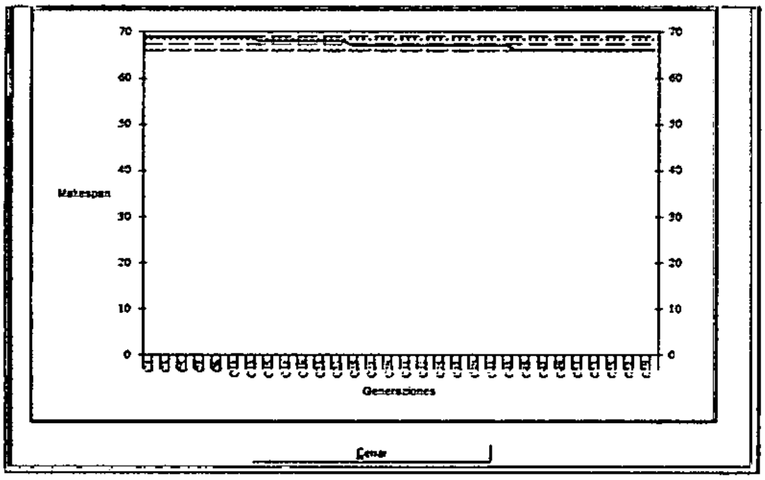


Figura N° 4.4: Comportamiento del algoritmo en la instancia N° 4, makespan de 66

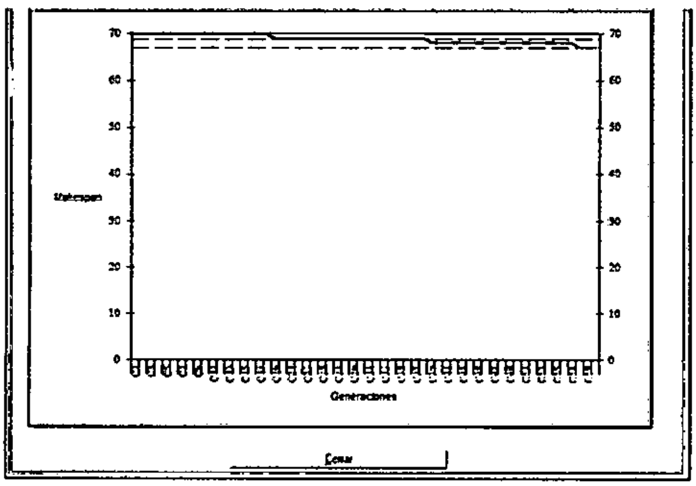


Figura N° 4.5: Comportamiento del algoritmo en la instancia N° 4, makespan de 67

<u>Instancia N° 5</u>: 5x10 (5 trabajos 10 máquinas)

Esta instancia en todas las corridas encuentra la solución óptima con el mismo makespan de 64, el coeficiente de variabilidad es de 0%, y para mayor

comodidad se muestra el Gantt, a través de una hoja de cálculo por la cantidad de máquinas y su comportamiento después de 20 generaciones con 10 individuos como tamaño de población, ver Figura N° 4.6

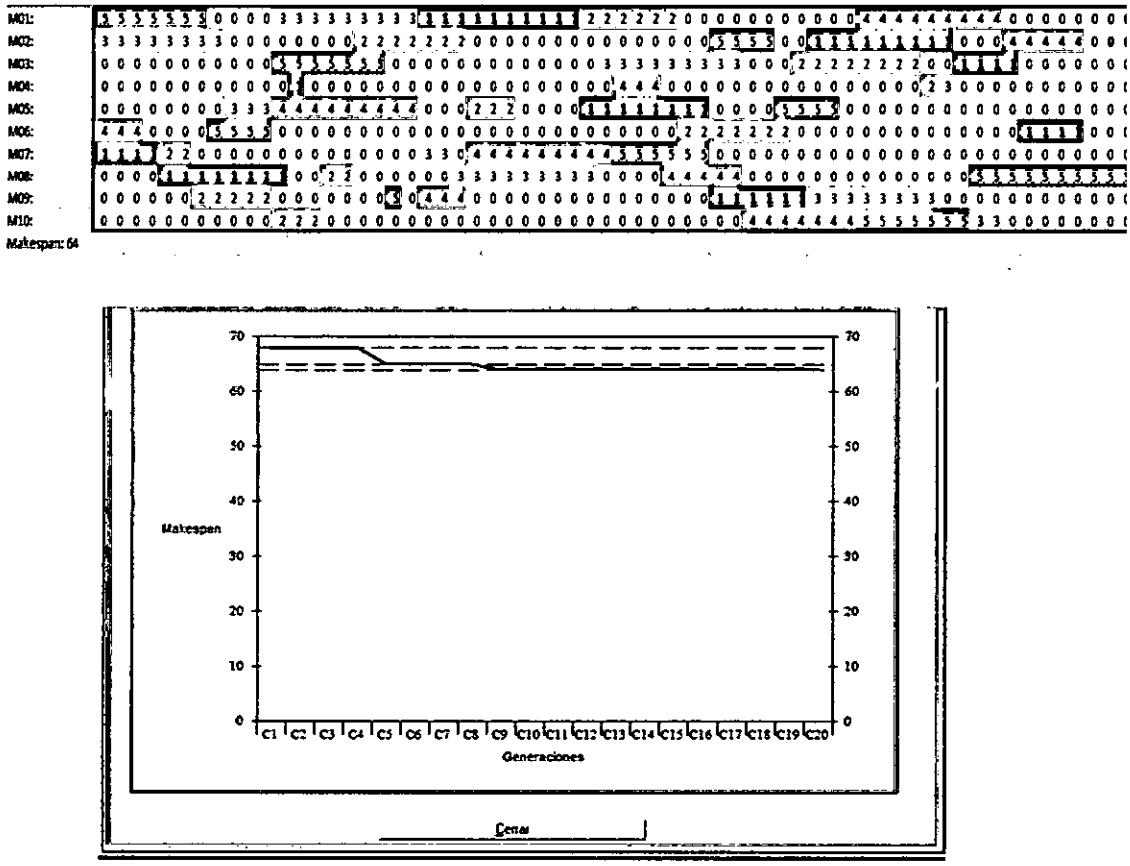


Figura N° 4.6: Gantt y comportamiento del algoritmo en la instancia N° 5

Instancia N° 6: 7x5 (7 trabajos 5 máquinas)

Las respuestas del algoritmo muestran que de 64 corridas 58 muestran un makespan de 52, 3 corridas muestran un makespan de 51 y 3 de 53; lo que hace un coeficiente de variabilidad de 0.59%, además los resultados muestran que no hay influencia estadística de ninguno de los parámetros en estudio como número de generaciones y el tamaño de la población, esta situación indica que el algoritmo demuestra ser altamente eficiente en el cálculo del makespan. Por otro lado se muestra que existen soluciones alternativas de las cuales se muestran 2 con su respectivo diagrama de Gantt y su comportamiento ver figura N° 4.7.

M01: 00 00 00 00 00 00 00 00 00 00 00 00 05 05 05 02 02 00 04 04 04 04 04 04 00 00 00 00 00 00 00 00 00 07 07 07 07 07 07 07 06 06 06 06 01 01 01 01 01 01 01 01 01 01
M02: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 07 07 07 05 05 05 05 03 00 06 06 06 06 06 04 04 04 04 04 04 04 01 01 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00
M03: 05 05 05 05 05 05 05 05 05 05 05 02 02 06 06 06 06 06 06 06 06 06 06 06 07 07 07 07 07 07 07 07 07 03 03 03 03 03 03 03 03 03 00 01 04 04 04 04 04 00 00 00 00 00
M04: 06 06 06 06 06 06 04 04 04 04 04 04 04 03 03 03 03 03 03 07 07 07 05 05 05 05 05 05 05 01 01 01 01 01 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
M05: 02 02 02 02 02 02 02 02 07 07 07 07 07 07 04 04 04 01 01 01 01 01 01 01 01 01 00 00 06 06 06 06 06 06 06 05 05 05 05 05 05 05 05 03 03 03 03 03 03 03 03 00
Makespan = 51

M01: 00 00 00 00 00 00 00 00 00 00 00 00 05 05 05 00 00 00 04 04 04 04 04 04 02 02 00 00 00 00 00 00 00 07 07 07 07 07 07 07 06 06 06 06 01 01 01 01 01 01 01 01 01 01
M02: 00 00 00 00 00 00 00 00 07 07 07 00 00 00 00 00 05 05 05 05 00 00 00 00 06 06 06 06 06 06 03 04 04 04 04 04 04 04 04 00 01 01 01 01 01 00 00 00 00 00 00 00 00 00
M03: 05 05 05 05 05 05 05 05 05 05 05 06 06 06 06 06 06 06 06 06 06 02 02 07 07 07 07 07 07 07 07 07 03 03 03 03 03 03 03 03 03 03 00 01 04 04 04 04 04 00 00 00 00 00
M04: 06 06 06 06 06 06 04 04 04 04 04 04 04 07 07 07 03 03 03 03 03 03 03 05 05 05 05 05 05 05 05 01 01 01 01 01 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
M05: 07 07 07 07 07 07 07 02 02 02 02 02 02 02 04 04 04 01 01 01 01 01 01 01 01 01 06 06 06 06 06 06 06 05 05 05 05 05 05 05 05 03 03 03 03 03 03 03 00 00 00
Makespan = 51

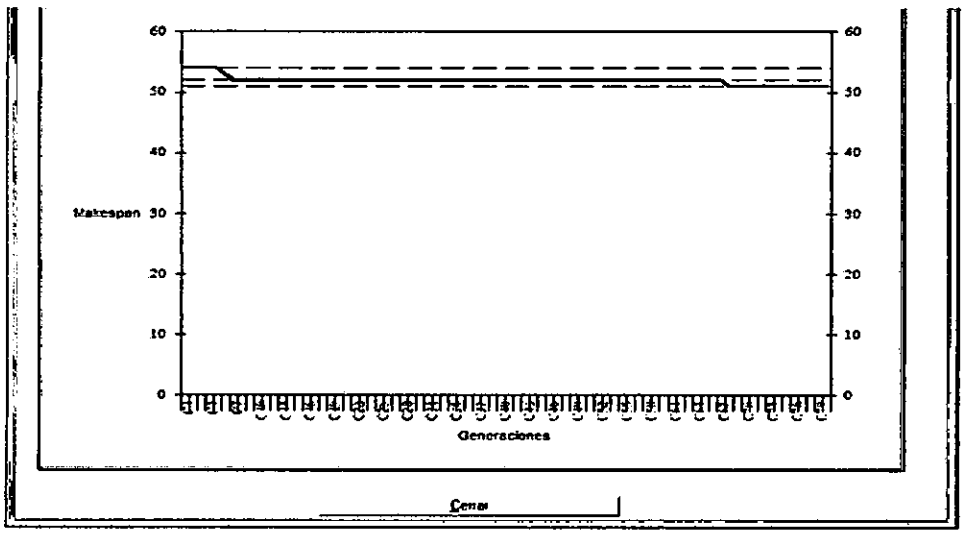


Figura N° 4.7: Gantt y comportamiento del algoritmo en instancia N°6

Instancia N° 7: 5x7 (5 trabajos 7 máquinas)

En todas las corridas del algoritmo, llega a la misma solución óptima, esto es un makespan de 56, el coeficiente de variabilidad es cero (0), y no hay influencia de los factores en estudio en el makespan; el diagrama de Gantt y el comportamiento del algoritmo se puede observar en el gráfico de la Figura N° 4.8, en el cual se observa que en la generación 9, alcanza la solución óptima:

M01: 04 04 04 04 04 04 04 00 00 00 00 00 00 00 00 00 00 00 00 00 02 02 00 00 00 00 00 00 00 00 03 03 03 05 05 05 05 05 05 05 01 01 01 01 01 01 01 01 00 00 00 00 00 00
M02: 00 00 00 00 00 00 00 00 00 04 04 00 00 00 00 00 00 00 00 00 00 00 02 00 00 05 05 05 05 05 05 05 05 00 00 03 03 03 03 03 03 03 00 00 00 00 00 00 00 00 01 01 00 00 00 00
M03: 00 00 00 00 00 00 00 00 00 00 00 00 00 03 03 03 03 03 03 03 03 03 03 05 00 00 00 00 00 00 00 00 00 00 04 04 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 01 01
M04: 03 03 05 05 05 05 05 05 05 05 05 01 01 01 01 01 01 01 01 01 00 00 02 02 02 02 02 02 00 00 00 00 00 00 00 00 00 04 04 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
M05: 02 02 02 02 00 00 00 00 00 00 00 00 05 05 05 05 05 05 05 00 00 00 03 03 03 03 04 04 04 04 04 04 04 01 01 01 01 01 01 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00
M06: 00 00 03 03 03 03 03 03 03 03 03 02 02 02 02 00 00 00 00 00 01 01 01 01 01 01 01 01 01 01 00 05 00 00 00 00 00 00 04 04 04 04 04 04 00 00 00 00 00 00 00 00 00 00
M07: 01 01 01 01 01 01 01 00 00 00 00 00 00 00 00 00 02 02 02 02 00 00 00 00 00 00 03 03 03 03 03 00 00 00 00 00 00 00 00 00 00 00 00 00 05 05 05 05 05 05 04 04 04 04 04 04 00 00
Makespan = 56

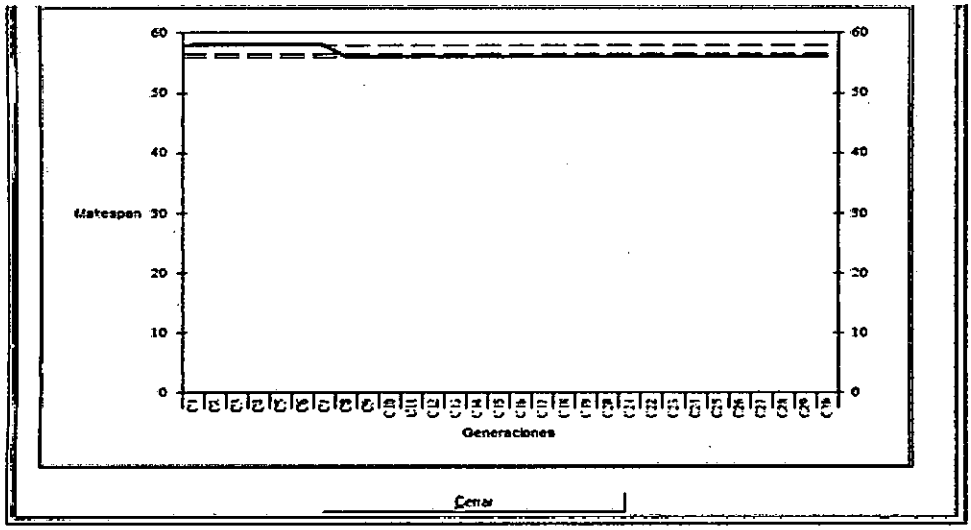


Figura N° 4.8: Gantt y comportamiento del algoritmo en instancia N°7

En Instancia N° 8: 6x6 (6 trabajos 6 máquinas)

En este caso el algoritmo encuentra la solución óptima con un makespan de 53 unidades en 6 corridas y las otras 58 corridas la solución muestra una unidad por encima del valor óptimo esto es un makespan de 54 unidades, conforme se muestra en el diagrama de Gantt; esto hace un coeficiente de variabilidad muy bajo del orden de 0.54% respecto a la media 53.91 además el análisis de varianza no muestra ningún efecto de los factores en estudio comportándose estadísticamente iguales el número de generaciones y el tamaño de la población. El gráfico de la Figura N° 4.9, muestra el comportamiento del algoritmo a lo largo de 100 generaciones se observa que aproximadamente en la generación 80 alcanza la solución óptima.

M01: 00 00 00 00 00 00 00 04 04 01 00 00 00 00 00 00 06 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 02 02 02 02 02 02 02 00 00 03 03 03 03 03 05 05 05 00 00 00 00 00 00
M02: 02 02 02 02 02 02 04 06 06 06 06 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 05 05 00 00 00 00 00 00 00 00 00 00 00 03 03 03 03 00 00 00 00 00
M03: 01 01 01 01 01 01 01 02 02 02 02 02 02 02 02 02 02 03 03 03 03 03 03 03 03 03 04 04 04 05 05 05 05 05 05 06 06 06 06 06 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
M04: 00 00 00 00 00 00 00 00 00 00 00 00 00 06 00 00 01 01 01 01 01 01 01 01 00 00 00 00 00 03 03 03 03 03 03 03 03 03 03 04 04 04 04 04 00 02 02 02 02 02 02 02 02 02 02 05 05 05 05 05 05
M05: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 02 02 02 02 02 02 02 02 02 06 06 06 06 06 01 01 01 01 05 00 00 04 04 04 04 04 00 00 00 00 00 00 00 00 00 00 00 03 00 00 00 00
M06: 00 00 00 00 00 00 00 00 00 00 00 00 00 06 06 06 00 00 00 00 00 00 00 01 01 01 01 01 01 02 02 02 02 00 00 00 03 03 03 03 03 03 05 05 05 05 04 04 04 04 04 04 00 00 00 00 00
Makespan = 54

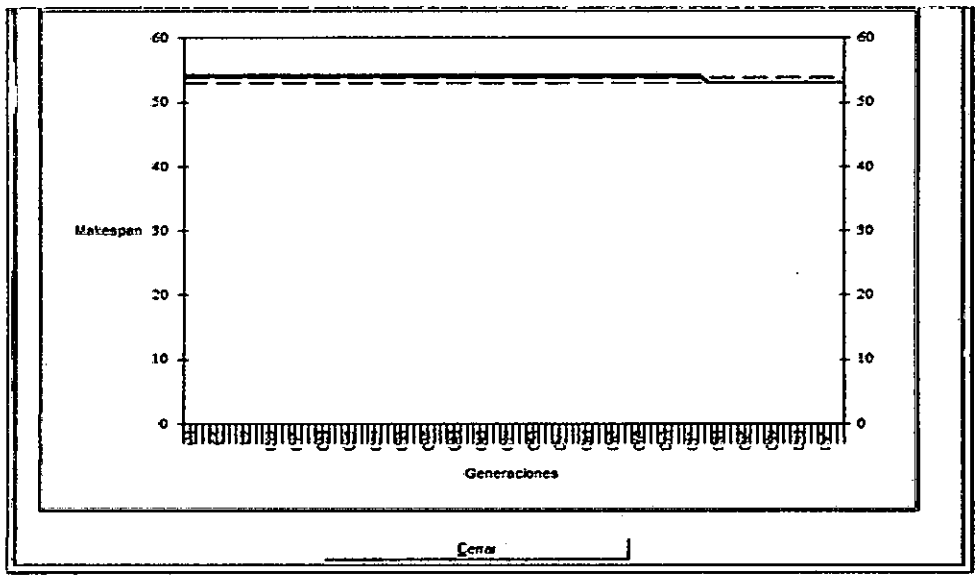


Figura N° 4.9: Gantt y comportamiento del algoritmo instancia N°8

En Instancia N° 9: 8x6 (8 trabajos 6 máquinas)

Esta instancia fue construida comenzando con la solución óptima y a partir de allí se reconstruyo la data de entrada para los tiempos de procesamiento en las máquinas y para la secuencia de las operaciones de los trabajos en las máquinas, en esta instancia el algoritmo genético logra en todas las corridas encontrar la solución óptima, con un makespan de 18, más aún existen soluciones alternativas. El coeficiente de variabilidad es de cero (0) y no existen diferencias estadísticas entre los factores. Según el gráfico de la Figura N° 4.10, la solución óptima se alcanza en la cuarta generación.

```
M01: 04 04 01 01 01 06 06 06 06 05 05 05 05 05 08 02 02 03
M02: 06 06 04 04 04 05 05 05 01 02 02 03 03 03 07 07 07 07
M03: 02 02 02 07 07 03 03 03 03 03 03 01 01 01 05 06 04 04
M04: 05 05 05 05 05 02 02 02 02 01 01 06 04 04 04 03 03 08
M05: 03 03 03 03 03 04 04 04 05 06 06 07 02 02 01 01 01 01
M06: 08 08 06 06 06 01 01 01 04 04 04 02 07 07 03 05 05 05
Makespan = 18
```

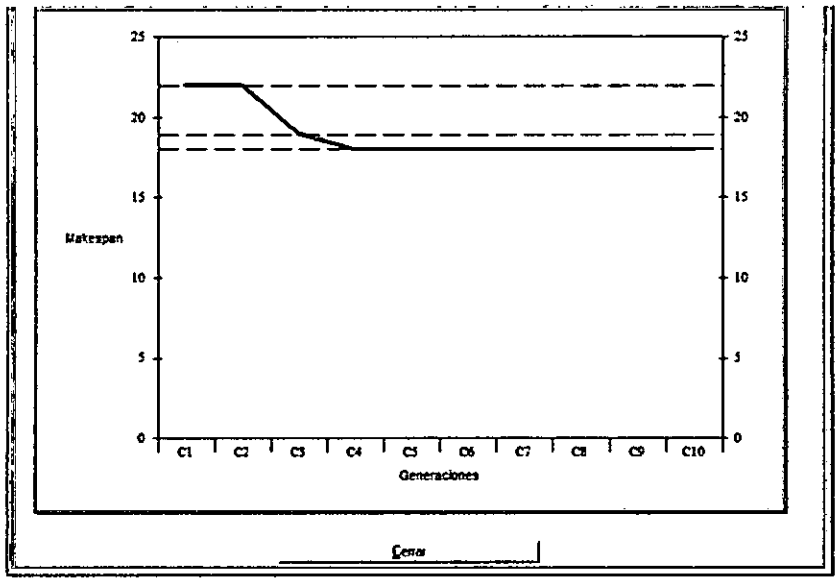


Figura N° 4.10: Gantt y comportamiento del algoritmo instancia N° 9.

Instancia N° 10: 13x10 (13 trabajos 10 máquinas)

Esta instancia de la misma manera que la anterior, fue construida comenzando con la solución óptima y a partir de allí se reconstruyo la data de entrada para los tiempos de procesamiento en las máquinas y para la secuencia de las operaciones de los trabajos en las máquinas, en esta instancia el algoritmo genético logra en todas las corridas encontrar la solución óptima, con un makespan de 16, y de la misma manera que en el caso anterior existen soluciones alternativas. El coeficiente de variabilidad es de cero (0) y no existen diferencias estadísticas entre los factores. Según el gráfico de la figura N° 4.11, la solución óptima se alcanza en la tercera generación.

```
M01: 01 01 01 01 01 01 01 01 01 05 05 05 05 05 05 05
M02: 02 02 02 02 02 02 02 06 06 06 06 06 06 06 06 06
M03: 10 10 10 10 11 09 09 09 09 09 09 09 09 09 13 13
M04: 03 03 03 03 03 03 04 11 11 11 08 08 08 08 08 08
M05: 04 04 04 07 07 07 07 07 07 07 07 02 02 02 02 02
M06: 05 05 05 05 05 05 05 05 05 01 01 01 01 01 01 01
M07: 06 06 06 06 06 06 06 04 04 04 04 07 07 07 07 07
M08: 07 07 11 11 10 10 10 10 10 10 10 10 10 04 04 04
M09: 08 08 08 08 08 08 08 08 08 08 03 03 03 03 03 03
M10: 09 09 09 09 09 11 11 02 02 02 02 12 12 12 12 12
Makespan = 16
```

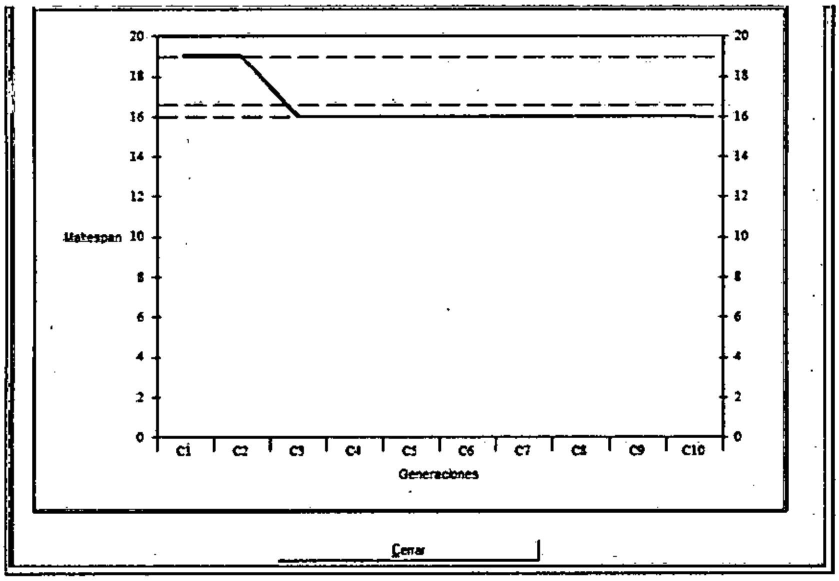


Figura N° 4.11: Gantt y comportamiento del algoritmo instancia N° 10.

En Instancia N° 11[2]: 10x2 (10 trabajos 2 máquinas)

Esta instancia fue tomada de internet, y la solución fue obtenida utilizando el algoritmo de Jackson, heurística que solo resuelve modelos de hasta 2 máquinas cómo máximo. Los resultados aplicando el algoritmo genético son coincidentes alcanzando un makespan de 68, en todas las corridas lo que muestra una alta eficiencia del algoritmo genético más aún el algoritmo genético advierte que hay múltiples soluciones, en este caso se muestra dos diagramas de Gantt y el comportamiento del algoritmo gráfico de la figura N° 4.12, muestra que se alcanza la solución óptima tan pronto se genera la solución de inicio y se mantiene hasta la última generación. Se vuelve a demostrar también que los parámetros en estudio no tienen ninguna influencia cuando el tamaño del problema en estudio es pequeño.

M01: 05 05 05 06 06 06 06 06 06 06 06 06 02 00 09 09 09 09 09 09 09 09 09 09 09 09 09 03 03 03 03 03 03 03 03 03 03 03 08 08 08 08 08 08 08 08 10 10 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
M02 01 01 01 01 01 01 01 01 09 09 09 09 09 09 03 03 03 03 03 03 03 03 03 03 03 03 03 04 08 08 08 09 08 08 08 08 08 08 06 06 05 06 06 06 06 06 07 07 07 07 07 07 10 10 10 10 10 10 02 02 02 02 02 02 02 02 02 02
Makespan = 68

M01: 10 10 06 06 06 05 05 06 06 06 06 02 01 01 01 05 05 05 08 08 08 08 08 08 08 08 00 00 00 00 00 00 00 00 00 00 00 00 03 03 03 03 03 03 03 03 03 03 03 03 09 09 09 09 09 09 09 09 09 09 09 09 00 00 00 00 00 00
M02 01 01 01 01 01 01 01 01 08 08 08 08 08 08 08 08 08 08 09 09 09 09 09 09 04 03 03 03 03 03 03 03 03 03 03 03 03 03 03 07 07 07 07 07 07 06 06 06 06 06 06 06 06 02 02 02 02 02 02 02 02 02 02 10 10 10 10 10 10
Makespan = 68

[2] http://cursos.tecmilenio.edu.mx/cursos/at8q3ozr5p/prof/io/io04005/anexos/explica10.pdf

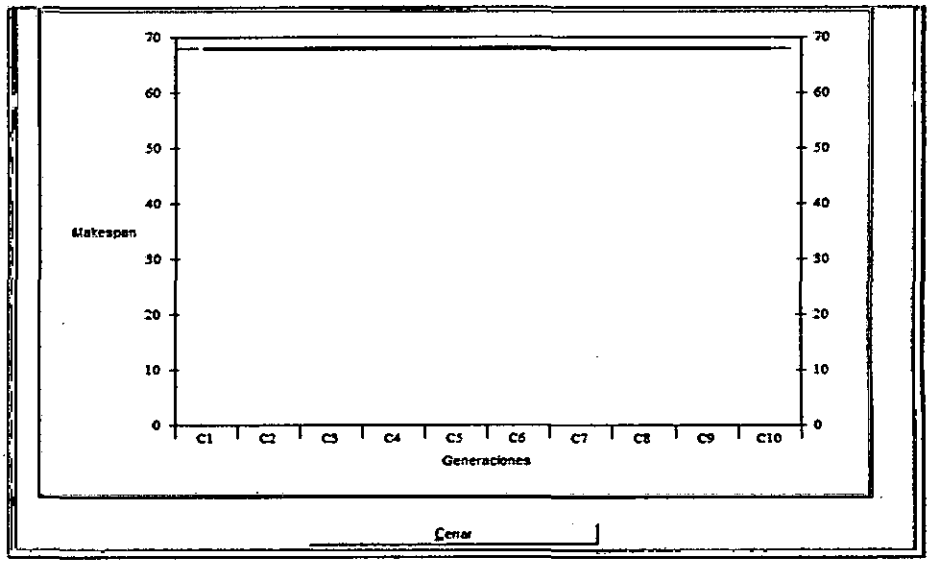


Figura N° 4.11: Gantt y comportamiento del algoritmo instancia N° 11.

4.3 ANÁLISIS Y DISCUSIÓN DE RESULTADOS

Los resultados muestran que el algoritmo en todas las instancias encuentra la solución óptima, y que no existe ningún efecto en la solución, de los factores en estudio "número de generaciones" y "tamaño de población"; así mismo se observa que la interacción de los dos factores tampoco tiene influencia en la solución, comportándose estadísticamente iguales.

El algoritmo no solo resuelve problemas del tipo Job Shop (problemas que tienen diferente secuencia de operación) sino también problemas del tipo Flow Shop (problemas que tienen la misma secuencia de operación).

Debe notarse que no todas las instancias requieren que los trabajos pasen por todas las máquinas, esta situación se ve reflejada en el tiempo de procesamiento de los trabajos en las máquinas con un valor igual a cero (0).

Dado que no existe influencia de los factores en estudio en el cálculo del makespan, un análisis integral concluye que de todas las instancias utilizadas para probar el algoritmo 8 de cada 11 encuentra la solución óptima con un coeficiente de variabilidad de 0% esto quiere decir que en todas las corridas

encuentra la solución óptima; en 1 de cada 11 instancias el 67.18% de las veces encuentra la solución óptima y el resto 32.82% encuentra soluciones 1.32% por encima del valor óptimo lo que hace un coeficiente de variabilidad del 1.19%; de la misma manera en 1 de cada 11 instancias existe un 4.68% de todas las corridas que encuentra la solución óptima y 90.62% de las veces encuentra soluciones 1.96% por encima del valor óptimo y el resto valores 3.92% por encima del valor optimo, este comportamiento hace que los resultados tengan un coeficiente de variabilidad del orden del 0.59% respecto de la media; finalmente en 1 instancia de cada 11, el 9.37% de las veces encuentra la solución óptima y 90.63% encuentra soluciones 1.89% por encima del valor óptimo, con un coeficiente de variabilidad del 0.54% respecto a la media. Estos resultados demuestran que el algoritmo es altamente eficiente en la solución del makespan para modelos de: Job Shop y Flow Shop.

Los operadores genéticos para los cuales el algoritmo funciona eficientemente son: 95% de porcentaje de cruzamiento, 5% de porcentaje de mutación, 60 individuos como tamaño de población y 80 generaciones.

Comparando estos resultados con los proporcionados V. Parada Daza, et al. (2011), quienes trabajando con un problema similar (Job Shop Flexible), implementaron un algoritmo genético para instancias similares en cuanto a tamaño del problema; en este caso para que el algoritmo funcione bien se requiere parámetros mucho mayores tales como: 4500 individuos como tamaño de población, 3000 generaciones, porcentaje de cruza 75% y 30% de porcentaje de mutación lo cual convierte al algoritmo, en una búsqueda aleatoria en lugar de una búsqueda evolutiva.

Finalmente el algoritmo para salidas del Diagrama de Gantt muy grandes, se ha implementado una salida a la hoja de cálculo de Excel para una mejor comprensión.

CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

1) El algoritmo genético implementado resuelve el problema de Job Shop Scheduling, en todas las pruebas realizadas cuando se refiere al cálculo del makespan, demostrándose la hipótesis planteada al inicio de la investigación, es decir que los algoritmos genéticos resuelven el problema del Job Shop Scheduling, con el añadido de que los buenos resultados se consiguen en un tiempo reducido de aproximadamente 55 segundos, para la instancia más grande.

2) Los parámetros del algoritmo que dieron buenos resultados son: 95% porcentaje de cruza y 5% porcentaje de mutación; 60 individuos en la población y 80 generaciones. Se ha verificado que valores superiores a estos parámetros no contribuyen a una mejora en el comportamiento de la solución.

3) El análisis de varianza en todos los casos mostraron que los factores en estudio (población inicial y número de generaciones), no tienen ninguna influencia en la solución del problema, así mismo se observó que la interacción de los factores tampoco ejerce ninguna influencia en los resultados del algoritmo, comportándose estadísticamente iguales.

4) Del análisis del comportamiento del algoritmo se observa que el tiempo de ejecución está directamente relacionado con el tiempo de procesamiento de los trabajos en las máquinas; se ha demostrado que instancias de 15 trabajos y 15 máquinas con tiempos de procesamiento que tienen un rango de variación de 0 a 100 el tiempo de ejecución del algoritmo es de 87 segundos, sin embargo la bondad de la solución se distancia en promedio 15.08% del mejor valor encontrado ver anexo N° 3. Luego para instancias del mismo tamaño esto es 15 trabajos y 15 máquinas con un rango de tiempo de procesamiento entre 0 y 10 el tiempo de ejecución es de 30 segundos; en todos los casos el tamaño de la población fue de 60 individuos y 80 generaciones.

5) El algoritmo implementado tiene las siguientes capacidades: Resolver el problema, capacidad de generar instancias en forma aleatoria con tiempos de procesamiento entre 0 y 10 unidades de tiempo, capacidad de graficar el comportamiento de la solución a lo largo de la ejecución del algoritmo y finalmente la salida de la solución puede imprimirse en pantalla o enviarse a una hoja de cálculo de MS Excel, para su análisis posterior.

5.2 RECOMENDACIONES

1) Como trabajo futuro, se recomienda implementar un algoritmo incorporando cruza dinámica, programación paralela y diseñar el algoritmo genético con edades para poder resolver problemas del mismo tamaño que los tratados aquí pero con tiempos de operación entre 0 y 100 unidades de tiempo y comparar los resultados con los proporcionados en el repositorio web.

2) Así mismo se recomienda hacer un sistema informático basado en el algoritmo genético que permita determinar la programación de trabajos a máquinas compactando la salida del Diagrama de Gantt cuando los tiempos de procesamiento son muy grandes.

6. BIBLIOGRAFÍA

Alba, E., & Troya, J. M. (2002). Improving flexibility and efficiency by adding parallelism to genetic algorithms. *Statistics and Computing, 12*(2), 91-114.

Alba, E., Luna, F., & Nebro, A. J. (2004). Advances in parallel heterogeneous genetic algorithms for continuous optimization. *International Journal of Applied Mathematics and Computer Science, 14*(3), 101-117.

Armentano, V. A., & Scrich, C. R. (2000). Tabu search for minimizing total tardiness in a job shop. *Int. J. Production Economics*(63), 131-140.

Bagchi Tapan, P. (1999). *MultiObjetive Scheduling by Genetic Algorithms.* New York: Kluwer Academic Publishers.

Blazewicz, J., Ecker, K. H., Schmidt, G., & Weglarz, J. (1994). Scheduling in Computer and manufacturing System. (7), 1-17.

Cortez Rivera, D. (2004). *Un Sistema Inmune Artificial para resolver el problema del Job Shop Scheduling.* Tesis de Maestría, Centro de investigación y estudios avanzados del Instituto Politécnico Nacional, Departamento de Ingeniería Eléctrica sección de computación.

García Martínez, C. (2008). *Algoritmos Genéticos Locales.* Tesis para optar el grado de Doctor, Universidad de Granada, Escuela Técnica Superior de Ingenierías Informática, Granada.

Glover, F., Kelly, J. P., & Laguna, M. (1995). Genetic algorithms and tabu search:hybrids for optimization. *Computers Ops Res.*(22), 111.

Gonçalves, J. F., & Beirão, N. C. (1999). Um algoritmo genético baseado em chaves aleatorias para sequenciamiento de operacoes. *Revista Associacao Portugesa de Desenvolvimento e Investigacao Operacional*(19), 123.

Gonçalves, J. F.; Mendes, J. J.; Resende, M. G. (2002). A Hybrid Genetic Algorithm for the Job Shop Scheduling Problem. *Technical Report TD-5EAL6J, AT&T Labs Research*(NJ 07932).

Jiménez Carrión, M. (2005). La técnica enumerativa de la Programación Dinámica en los Problemas de Producción e Invesntario y los Algoritmos Genéticos. *VI Congreso Chileno de Investigación de Operaciones.* Valdivia.

Matsuo, H., Suh, C. J., & Sullivan, R. S. (1988). *A controlled search simulated annealing method for the general jobshop scheduling problem.* In Working Paper, Graduate School of Business, University of Texas, Austin.

Medina Duran, R., Pradenas Rojas, L., & Parada Daza, V. (2011). Un algoritmo genético para el problema de Job Shop Flexible. *Ingeniare. Revista chilena de ingeniería, 19*(1), pp.53-61.

Merkle, D., & Middendorf, M. (2002). Ant colony optimization with the relative pheromone evaluation method. (S. Cagnoni, J. Gottlieb, E. Hart, M. Middenford, & G. Raidl, Edits.) *Applications of Evolutionary Computing, Proceedings of EvoWorkshops, 2279*, 321.

Peña, V., & Zumelzu, L. (2006). *Estado del Arte del Job Shop Scheduling Problem*. Obtenido de www.alumnos.inf.utfsm.cl/~vpena/ramos/ili295/ia-obshop.pd

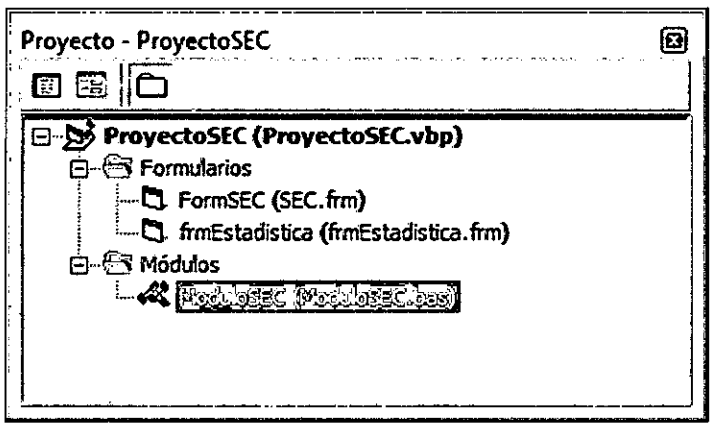
Teich, T., Fischer, M., Vogel, A., & Fischer, J. (2001). Solving the job shop scheduling problem using ant algorithms. *In Proceedings of the 17th International Conference on CAD/CAM, Robotics and Factories of Future* (pág. 604). Durban, South Africa: Springer-Verlag.

ANEXOS

ANEXO Nº 1

CÓDIGO DEL ALGORITMO GENÉTICO

La estructura del algoritmo desarrollado en Visual Basic 6.0 es la siguiente:



Donde:

ProyectoSec.vbp.- Representa el nombre del proyecto Secuenciación y está compuesto por dos formularios y un módulo de programa.
SEC.frm.- Es un archivo que contiene el formulario que sirve de interface entre el usuario y la máquina.
frmEstadística.frm.- Archivo que contiene el formulario que se utiliza para graficar el comportamiento del algoritmo durante su ejecución.
ModuloSEC.bas.- Archivo que contiene el módulo de programación del Algoritmo genético.

A continuación se presenta el módulo completo que se inicia con una parte declarativa de variables de la siguiente manera:

SECCIÓN DE DECLARACIÓN DE VARIABLES

```
Public Problem As Problema
Public M() As Integer
Public tiempoMaximo As Integer
Public TPo As Integer
Public po As Integer
Public fil As Integer
Public col As Integer
Public Indiv() As Integer
Public NG As Integer
Public contieneMak() As Integer
Public NombreDelProblema As String
Public tareas() As jobs
Public LineTiempo() As Integer
Public maq As Integer
Public ind As Integer
Public trab As Integer
Public h As Integer, l1 As Integer, cr1 As Integer, cr2 As Integer
Public deco1() As Integer
```

```
Public deco2() As Integer
Public z As Integer
Public q As Integer
Public eLite As Integer
Public meJorp As Integer
Public indPeor As Integer
Public peOrp As Integer
Public mejorIndividuo() As Integer
Public mejorFitness As Integer
Public mejorMakespan As Integer
Public pMuta As Single
```

ESTRUCTURA E DATOS DISEÑADOS PARA EL ALGORITMO

```
Type Problema
   NumTrabajos As Integer
   NumMaquinas As Integer
   Secuencia() As Integer
   Tiempo() As Integer
End Type

Type Individuo
  operacion() As Integer
  fitness As Integer
  makespan As Integer
End Type

Type jobs
  trabajos As Integer
  secuen As String
  ind As Integer
  secuen1 As String
End Type

Type generacion
   poblacion() As Individuo
End Type

Public ecosistema As generacion

Public padres As generacion
```

PROCEDIMIENTO QUE GENERA PROBLEMAS ALEATORIAMENTE

```
Public Sub GeneraProblema()
  Problem.NumTrabajos = InputBox("Ingrese el Número de Trabajos", "Algoritmo Evolutivo
SEC")
  Problem.NumMaquinas = InputBox("Ingrese el Número de Máquinas", "Algoritmo
Evolutivo SEC")
  ReDim Problem.Secuencia(Problem.NumTrabajos - 1, Problem.NumMaquinas - 1)
  ReDim Problem.Tiempo(Problem.NumMaquinas - 1, Problem.NumTrabajos - 1)
  ReDim M(Problem.NumMaquinas - 1, Problem.NumTrabajos - 1)

  Randomize
  For i = 0 To Problem.NumMaquinas - 1
    For j = 0 To Problem.NumTrabajos - 1
      Problem.Tiempo(i, j) = Rnd * 10
      M(i, j) = -1
    Next j
  Next i

  For j = 0 To (Problem.NumTrabajos - 1)
    inicio = Int(Rnd * (Problem.NumMaquinas)) 'máquina de inicio
    M(0, j) = inicio
      For i = 1 To (Problem.NumMaquinas - 1)
        siguientemaquina = Int(Rnd * (Problem.NumMaquinas - 1))
        For checa = 0 To Problem.NumMaquinas - 1
          If M(checa, j) = siguientemaquina Then
            siguientemaquina = siguientemaquina + 1
            If siguientemaquina >= Problem.NumMaquinas Then
              siguientemaquina = 0
            End If
            checa = -1
          End If
        Next checa
        M(i, j) = siguientemaquina
      Next i
  Next j

  For i = 0 To Problem.NumTrabajos - 1
    For j = 0 To Problem.NumMaquinas - 1
        Problem.Secuencia(i, j) = M(j, i)
    Next j
  Next i

End Sub
```

PROCEDIMIENTO QUE GENERA UNA POBLACIÓN DE INDIVIDUOS

```
Public Sub GeneraPoblacionInicial()
  TPo = InputBox("Ingrese el tamaño de la población", "Algoritmo SEC")
  ReDim ecosistema.poblacion(TPo - 1)
  ReDim padres.poblacion(TPo - 1)
  ReDim M(Problem.NumMaquinas - 1, Problem.NumTrabajos - 1)
  For i = 0 To TPo - 1
    ReDim ecosistema.poblacion(i).operacion(Problem.NumTrabajos *
Problem.NumMaquinas - 1)
    ReDim padres.poblacion(i).operacion(Problem.NumTrabajos * Problem.NumMaquinas
- 1)
  Next i
  ReDim mejorIndividuo(Problem.NumTrabajos * Problem.NumMaquinas - 1)
  For i = 0 To TPo - 1
    For a = 0 To Problem.NumMaquinas - 1
      For b = 0 To Problem.NumTrabajos - 1
        M(a, b) = -1
      Next b
    Next a
    For j = 0 To (Problem.NumMaquinas - 1)
      Randomize
      inicio = Int(Rnd * (Problem.NumTrabajos)) 'trabajo de inicio
      M(j, 0) = inicio
      For a = 1 To (Problem.NumTrabajos - 1)
        Randomize
        siguienteTrabajo = Int(Rnd * (Problem.NumTrabajos - 1))
        For checa = 0 To Problem.NumTrabajos - 1
          If M(j, checa) = siguienteTrabajo Then
            siguienteTrabajo = siguienteTrabajo + 1
            If siguienteTrabajo >= Problem.NumTrabajos Then
              siguienteTrabajo = 0
            End If
            checa = -1
          End If
        Next checa
        M(j, a) = siguienteTrabajo
      Next a
    Next j
    k = 0
    For a = 0 To Problem.NumTrabajos - 1
      For b = 0 To Problem.NumMaquinas - 1
        ecosistema.poblacion(i).operacion(k) = M(b, a)
        k = k + 1
      Next b
    Next a
  Next i

End Sub
```

PROCEDIMIENTO QUE EVALUA A INDIVIDUOS Y CALCULA EL MAKESPAN

```
Public Sub EvaluaOperacion()
  fil = Problem.NumMaquinas
  calccol
  ReDim tareas(Problem.NumTrabajos - 1)
  ReDim LineTiempo(fil, col)
  For p = 0 To Problem.NumTrabajos - 1
    For j = 0 To Problem.NumMaquinas - 1
      tareas(p).secuen = tareas(p).secuen & CStr(Problem.Secuencia(p, j)) & ","
    Next j
  Next p
  For po = 0 To TPo - 1
    '**********************************
    ReDim LineTiempo(fil, col)
    mks = 0
    For p = 0 To Problem.NumTrabajos - 1
      tareas(p).secuen1 = tareas(p).secuen
      tareas(p).trabajos = 0
    Next p
    '**********************************
    For i = 0 To Problem.NumMaquinas * Problem.NumTrabajos - 1
      nt = ecosistema.poblacion(po).operacion(i)
      ini = 1
      t = tareas(nt).secuen1
      '*****************************************
      Do While True
        If CStr(Right(Left(t, ini), 1)) = "," Then
            mcp = CInt(Left(t, ini - 1))
            t = Right(t, Len(t) - ini)
            tareas(nt).secuen1 = t
            Exit Do
        End If
        ini = ini + 1
      Loop
      '*****************************************
      Do While True
        indtrab = tareas(nt).trabajos
        duracion = 0
        dt = Problem.Tiempo(mcp, nt)
        For x = indtrab To col
          If duracion = dt Then
            Exit Do
          End If
          If LineTiempo(mcp, x) = 0 Then
            duracion = duracion + 1
          Else
            If duracion < dt Then
              duracion = 0
            End If
          End If
          indtrab = indtrab + 1
```

```
            Next x
          Loop
          tareas(nt).trabajos = indtrab - dt
          For j = tareas(nt).trabajos To tareas(nt).trabajos + dt - 1
            LineTiempo(mcp, j) = nt + 1
          Next j
          tareas(nt).trabajos = tareas(nt).trabajos + dt
          If mks < tareas(nt).trabajos Then
            mks = tareas(nt).trabajos
          End If
        Next i
        ecosistema.poblacion(po).makespan = mks
      Next po
End Sub
```

PROCEDIMIENTO QUE CALCULA EL FITNESS DE CADA INDIVIDUO

```
Public Sub calculaFitness()
  For i = 0 To TPo - 1
    ecosistema.poblacion(i).fitness = tiempoMaximo - ecosistema.poblacion(i).makespan
  Next i
End Sub
```

PORCEIMIENTO PARA SELECCIONAR A LOS MEJORES INDIVIDUOS

```
Public Sub Ruleta()
  Dim VRuleta As Double, SParcial As Double, SAptitud As Double
  Dim elegido As Integer
  SAptitud = 0
  For i = 0 To TPo - 1
    SAptitud = SAptitud + ecosistema.poblacion(i).fitness
  Next i
  For i = 0 To TPo - 1
    SParcial = 0
    elegido = 0
    Randomize
    VRuleta = Rnd * SAptitud
    Do
      SParcial = SParcial + ecosistema.poblacion(elegido).fitness
      If SParcial >= VRuleta Then
        For j = 0 To Problem.NumMaquinas * Problem.NumTrabajos - 1
          padres.poblacion(i).operacion(j) = ecosistema.poblacion(elegido).operacion(j)
        Next j
        padres.poblacion(i).fitness = ecosistema.poblacion(elegido).fitness
        padres.poblacion(i).makespan = ecosistema.poblacion(elegido).makespan
      Else
        elegido = elegido + 1
      End If
    Loop While (SParcial < VRuleta)
  Next i
End Sub
```

PROCEDIMIENTO PARA REPRODUCCIÓN DE LOS INDIVIDUOS SELECCIONADOS

```
Public Sub reproduccion()

  ReDim deco1(Problem.NumMaquinas - 1, Problem.NumTrabajos - 1)
  ReDim deco2(Problem.NumMaquinas - 1, Problem.NumTrabajos - 1)

  'for que recorre la mitad de la población
  For h = 0 To (TPo / 2 - 1)
    Randomize
    If Rnd <= 0.95 Then   '0.95

      'estos dos For es para decodificar los individuos que son solución del problema
      For i = 0 To Problem.NumMaquinas - 1
        For j = 0 To Problem.NumTrabajos - 1
          deco1(i, j) = padres.poblacion(2 * h).operacion(Problem.NumMaquinas * j + i)
          deco2(i, j) = padres.poblacion(2 * h + 1).operacion(Problem.NumMaquinas * j +
i)
        Next j
      Next i

      'este For es para recorrer todas las máquinas con trabajos que no se repiten
      For q = 0 To Problem.NumMaquinas - 1
        'seleciona puntos de cruce
        cr1 = Round((Problem.NumTrabajos - 1) * Rnd, 0)
        cr2 = Round((Problem.NumTrabajos - 1) * Rnd, 0)
        If cr1 = cr2 Then
          cr1 = Int(Problem.NumTrabajos / 4)
          cr2 = Int(3 * Problem.NumTrabajos / 4)
        End If
        If cr2 < cr1 Then
          z = cr1
          cr1 = cr2
          cr2 = z
        End If

        'For que intercambia el material genético entre los puntos seleccionados
        For z = cr1 To cr2
          l1 = deco1(q, z)
          deco1(q, z) = deco2(q, z)
          deco2(q, z) = l1
        Next z

        'For que permite completar el intercambio genético del Hijo 1
        For i = 0 To Problem.NumTrabajos - 1
          If i = cr1 Then
            i = cr2 + 1
          End If
          If i < Problem.NumTrabajos Then
            j = cr1
            Do While j <= cr2
              If deco1(q, i) = deco1(q, j) Then
```

```
                deco1(q, i) = deco2(q, j)
                j = cr1
              Else
                j = j + 1
              End If
            Loop
          End If
        Next i

        'For que permite completar el intercambio genético para el Hijo 2
        For i = 0 To Problem.NumTrabajos - 1
          If i = cr1 Then
            i = cr2 + 1
          End If
          If i < Problem.NumTrabajos Then
            j = cr1
            Do While j <= cr2
              If deco2(q, i) = deco2(q, j) Then
                deco2(q, i) = deco1(q, j)
                j = cr1
              Else
                j = j + 1
              End If
            Loop
          End If
        Next i
      Next q
      'For que permite codificar nuevamente los padres despues del cruzamiento
      For i = 0 To Problem.NumMaquinas - 1
        For j = 0 To Problem.NumTrabajos - 1
          padres.poblacion(2 * h).operacion(Problem.NumMaquinas * j + i) = deco1(i, j)
          padres.poblacion(2 * h + 1).operacion(Problem.NumMaquinas * j + i) = deco2(i, j)
        Next j
      Next i
    End If
  Next h

'Crear a partir de los padres la nueva generación
For h = 0 To TPo - 1
  For a = 0 To Problem.NumMaquinas * Problem.NumTrabajos - 1
    ecosistema.poblacion(h).operacion(a) = padres.poblacion(h).operacion(a)
  Next a

Next h
'checkin

End Sub
```

PROCEDIMIENTO DE MUTACIÓN DE INDIVIDUOS

```
Public Sub Mutacion()
  Dim aleatorio As Single
  Dim a As Single, a1 As Single, a2 As Single
  Dim temporal As Integer
  Dim p1 As Integer
  Dim p2 As Integer
  pMuta = 0.05 '.05
  For j = 0 To TPo - 1
    Randomize
    aleatorio = Rnd
    If aleatorio < pMuta Then
      a = Rnd
      a1 = Rnd
      a2 = Rnd
      p1 = Problem.NumMaquinas * Round((Problem.NumTrabajos - 1) * a1, 0) +
Round((Problem.NumMaquinas - 1) * a, 0)
      p2 = Problem.NumMaquinas * Round((Problem.NumTrabajos - 1) * a2, 0) +
Round((Problem.NumMaquinas - 1) * a, 0)
      temporal = ecosistema.poblacion(j).operacion(p1)
      ecosistema.poblacion(j).operacion(p1) = ecosistema.poblacion(j).operacion(p2)
      ecosistema.poblacion(j).operacion(p2) = temporal
    End If
  Next
End Sub
```

PROCEDIMIENTO QUE BUSCA AL PEOR INDIVIDUO DE UNA POBLACIÓN

```
Public Sub peOr()
  indPeor = 0
  peOrp = ecosistema.poblacion(0).fitness

  For i = 0 To TPo - 1
    If peOrp > ecosistema.poblacion(i).fitness Then
      peOrp = ecosistema.poblacion(i).fitness
      indPeor = i
    End If
  Next i
End Sub
```

PROCEDIMIENTO QUE PERMITE REEMPLAZAR EL PEOR INDIVIDUO DE UNA POBLACIÓN POR EL MEJOR INDIVIDUO DE LA POBLACIÓN ANTERIOR.

```
Public Sub RemplazaPxM()
  For z = 0 To Problem.NumMaquinas * Problem.NumTrabajos - 1
      ecosistema.poblacion(indPeor).operacion(z) = mejorIndividuo(z)
  Next z
  ecosistema.poblacion(indPeor).fitness = mejorFitness
  ecosistema.poblacion(indPeor).makespan = mejorMakespan
End Sub
```

PROCEDIMIENTO QUE BUSCA AL MEJOR INDIVIDUO DE UNA POBLACIÓN

```
Public Sub mejor()
  eLite = 0
  meJorp = 0
  For i = 0 To TPo - 1
    If meJorp < ecosistema.poblacion(i).fitness Then
      meJorp = ecosistema.poblacion(i).fitness
      eLite = i
    End If
  Next i
  For i = 0 To Problem.NumMaquinas * Problem.NumTrabajos - 1
    mejorIndividuo(i) = ecosistema.poblacion(eLite).operacion(i)
  Next i
  mejorFitness = ecosistema.poblacion(eLite).fitness
  mejorMakespan = ecosistema.poblacion(eLite).makespan

End Sub
```

PROCEDIMIENTO QUE CONCENTRA TODOS LOS PROCEDIMIENTOS (ALGORITMO), PARA ENCONTRAR LA SOLUCIÓN A UN PROBLEMA

```
Public Sub Automatico()
  GeneraPoblacionInicial
  NG = InputBox("Ingrese el número de generaciones", "Algoritmo SEC")
  ReDim contieneMak(NG - 1)
  EvaluaOperacion
  maximoTiempo
  calculaFitness
  For y = 0 To NG - 1
    mejor
    'contieneMak(y) = ecosistema.poblacion(eLite).makespan
    Ruleta
    reproduccion
    Mutacion
    EvaluaOperacion
    calculaFitness
    peOr
    RemplazaPxM
    MsgBox ("Generacion " & CStr(y) & "lista")
  Next y
  MsgBox ("listo")
End Sub
```

PROCEIMIENTO QUE PERMITE INGRESAR DATOS DE UN PROBLEMA

```
Public Sub IngresarProblema()
  Problem.NumTrabajos = InputBox("Ingrese el Número de Trabajos", "Algoritmo Evolutivo
SEC")
  Problem.NumMaquinas = InputBox("Ingrese el Número de Máquinas", "Algoritmo
Evolutivo SEC")
  ReDim Problem.Secuencia(Problem.NumTrabajos - 1, Problem.NumMaquinas - 1)
  ReDim Problem.Tiempo(Problem.NumMaquinas - 1, Problem.NumTrabajos - 1)
  ReDim M(Problem.NumMaquinas - 1, Problem.NumTrabajos - 1)

  For i = 0 To Problem.NumMaquinas - 1
    For j = 0 To Problem.NumTrabajos - 1
      Problem.Tiempo(i, j) = InputBox("Ingrese el tiempo en la máquina " & i & " para el
trabajo " & j, "Algoritmo Evolutivo SEC")
    Next j
  Next i

  For i = 0 To (Problem.NumTrabajos - 1)
    For j = 0 To (Problem.NumMaquinas - 1)
        M(j, i) = InputBox("Ingrese el Nº de máq. " & j & " para el Trabajo " & i, "Algoritmo
Evolutivo SEC")
    Next j
  Next i

  For i = 0 To Problem.NumTrabajos - 1
    For j = 0 To Problem.NumMaquinas - 1
        Problem.Secuencia(i, j) = M(j, i)
    Next j
  Next i

End Sub
```

PROCEDIMIENTO QUE DETERMINA EL MÁXIMO TIEMPO DEL MAKESPAN

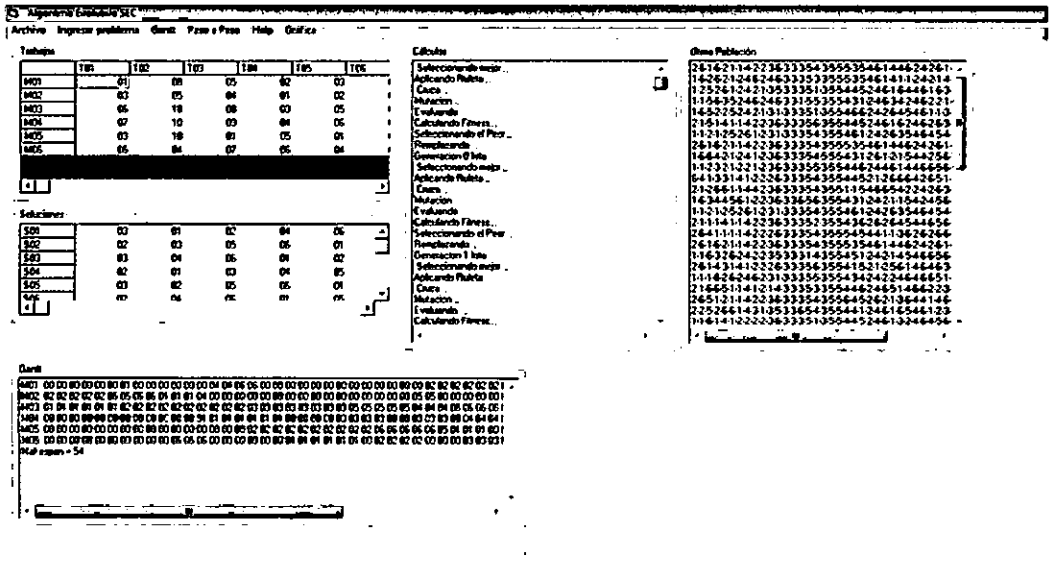
```
Public Sub maximoTiempo()
  tiempoMaximo = 0
  For i = 0 To Problem.NumMaquinas - 1
    For j = 0 To Problem.NumTrabajos - 1
      tiempoMaximo = tiempoMaximo + Problem.Tiempo(i, j)
    Next j
  Next i

  'For p = 0 To TPo - 1
  ' If tiempoMaximo < ecosistema.poblacion(p).makespan Then
  '   tiempoMaximo = ecosistema.poblacion(p).makespan
  ' End If
  'Next p
End Sub
```

PROCEDIMIENTO QUE MUESTRA LA SOLUCIÓN DEL PROBLEMA COMO UN DIAGRAMA DE GANTT

```
Public Sub SolucionGantt()
  fil = Problem.NumMaquinas
  calccol
  ReDim tareas(Problem.NumTrabajos - 1)
  ReDim LineTiempo(fil, col)

  For p = 0 To Problem.NumTrabajos - 1
    For j = 0 To Problem.NumMaquinas - 1
      tareas(p).secuen = tareas(p).secuen & CStr(Problem.Secuencia(p, j)) & ","
    Next j
  Next p
  mks = 0
  For p = 0 To Problem.NumTrabajos - 1
    tareas(p).secuen1 = tareas(p).secuen
    tareas(p).trabajos = 0
  Next p
    '**********************************
  For i = 0 To Problem.NumMaquinas * Problem.NumTrabajos - 1
    nt = ecosistema.poblacion(eLite).operacion(i)
    ini = 1
    t = tareas(nt).secuen1
    '*****************************************
    Do While True
      If CStr(Right(Left(t, ini), 1)) = "," Then
        mcp = CInt(Left(t, ini - 1))
        t = Right(t, Len(t) - ini)
        tareas(nt).secuen1 = t
        Exit Do
      End If
      ini = ini + 1
    Loop
      '*****************************************
    Do While True
      indtrab = tareas(nt).trabajos
      duracion = 0
      dt = Problem.Tiempo(mcp, nt)
      For x = indtrab To col
        If duracion = dt Then
          Exit Do
        End If
        If LineTiempo(mcp, x) = 0 Then
          duracion = duracion + 1
        Else
          If duracion < dt Then
            duracion = 0
          End If
        End If
        indtrab = indtrab + 1
      Next x
```

```
    Loop
    tareas(nt).trabajos = indtrab - dt
    For j = tareas(nt).trabajos To tareas(nt).trabajos + dt - 1
      LineTiempo(mcp, j) = nt + 1
    Next j
    tareas(nt).trabajos = tareas(nt).trabajos + dt
    If mks < tareas(nt).trabajos Then
      mks = tareas(nt).trabajos
    End If
  Next i
End Sub
```

EL FORMULARIO QUE HACE DE INTERFACE ENTRE EL USUARIO Y EL ORDENADOR ES:



EL CÓDIGO QUE RESPALDA ESTE FORMULARIO ES:

```
Private Sub Command1_Click()
  Dim x As String
  Dim y As String
  Print ""
  GeneraPoblacionInicial
  x = ""
  y = ""
  For i = 0 To TPo - 1
    k = 0
    For j = 0 To Problem.NumMaquinas * Problem.NumTrabajos - 1
      x = x & ecosistema.poblacion(i).operacion(k) + 1 & " "
      k = k + 1
    Next j
```

```
      Print x
      Print y
      Print ""
      x = ""
      y = ""
    Next i
End Sub



Private Sub AbreArchTex_Click()
  Cls
  Dim LineaDeTexto As String
  With dlgCommondialog
    .InitDir = App.Path
    .DialogTitle = "Abrir"
    .CancelError = False
    .Filter = "Archivos SEC (*.sec)|*.sec"
    .CancelError = True
    On Error GoTo errhandler
    .ShowOpen
    If Len(.FileName) = 0 Then
      Exit Sub
    End If

    sFile = .FileName
    NombreDelProblema = sFile
  End With

  MousePointer = 11
  Open dlgCommondialog.FileName For Input As #1
  Line Input #1, LineaDeTexto$
  Problem.NumMaquinas = CInt(LineaDeTexto$)
  Line Input #1, LineaDeTexto$
  Problem.NumTrabajos = CInt(LineaDeTexto$)
  ReDim Problem.Tiempo(Problem.NumMaquinas - 1, Problem.NumTrabajos - 1)
  ReDim Problem.Secuencia(Problem.NumTrabajos - 1, Problem.NumMaquinas - 1)
  ReDim M(Problem.NumMaquinas - 1, Problem.NumTrabajos - 1)

  For i = 0 To Problem.NumMaquinas - 1
    Line Input #1, LineaDeTexto$
    a = 1
    For j = 0 To Problem.NumTrabajos - 1
      Problem.Tiempo(i, j) = CInt(Trim(Mid(LineaDeTexto$, a, 3)))
      a = a + 3
    Next j
  Next i

  For i = 0 To Problem.NumTrabajos - 1
    Line Input #1, LineaDeTexto$
```

```
    a = 1
    For j = 0 To Problem.NumMaquinas - 1
      Problem.Secuencia(i, j) = CInt(Trim(Mid(LineaDeTexto$, a, 3)))
      a = a + 3
    Next j
  Next i


  Close #1
  MousePointer = 0

  Me.Caption = "Algoritmo Evolutivo SEC " + sFile
errhandler:
  Exit Sub
End Sub


Private Sub Command2_Click()
  EvaluaOperacion
End Sub

Private Sub estaDist_Click()
  Load frmEstadistica
  frmEstadistica.Show vbModal
End Sub

Private Sub Gantt_Click()
  mejor
  SolucionGantt
  rspta = InputBox("precione 1. para salida por pantalla 2. para salida a excel")
  If rspta = 1 Then
    '*********************************************************
    For i = 0 To fil - 1
      For j = 0 To ecosistema.poblacion(eLite).makespan - 1
        x = x & Format(LineTiempo(i, j), "00") & " "
      Next j
      Print "M" & Format(i + 1, "00") & ": " & x
      x = ""
    Next i

    Print "Makespan = " & ecosistema.poblacion(eLite).makespan
    Print ""
    '*********************************************************


    For i = 0 To fil
      For j = 0 To col
        LineTiempo(i, j) = 0
      Next j
    Next i
```

```
    Else
        'salida por hoja de calculo
        Dim objExcel As Object

        'Start a new workbook in Excel
        Set objExcel = CreateObject("Excel.Application")
        objExcel.Visible = False
        objExcel.Workbooks.Add
        objExcel.Columns("A").ColumnWidth = 14

        'Add data to cells of the first worksheet in the new workbook
        Set hoja = objExcel.ActiveSheet

        For i = 0 To fil - 1
            For j = 0 To ecosistema.poblacion(eLite).makespan - 1
                hoja.Cells(i + 1, j + 2).Value = LineTiempo(i, j)
            Next j
            hoja.Range("A" & i + 1) = "M" & Format(i + 1, "00") & ": "
        Next i
        hoja.Range("B:BFH").ColumnWidth = 2
        hoja.Range("A" & i + 1) = "Makespan: " & ecosistema.poblacion(eLite).makespan

        'ponemos visible
        objExcel.Visible = True

    End If

End Sub


Private Sub genProb_Click()
    Cls
    GeneraProblema
End Sub


Private Sub Command10_Click()
    mejor
    x = ""
    'For i = 0 To Problem.NumMaquinas * Problem.NumTrabajos - 1
    '   x = x & mejorIndividuo(i)
    'Next i
    'Print ""

    'Print x & mejorFitness & mejorMakespan

End Sub

Private Sub Command11_Click()
    Mutacion
```

```
End Sub

Private Sub Command12_Click()
  peOr
End Sub

Private Sub Command13_Click()
  RemplazaPxM
End Sub

Private Sub Command14_Click()
  calculaFitness
End Sub

Private Sub IngresaProblema_Click()
  Cls
  IngresarProblema
End Sub

Private Sub muestraProb_Click()
  Cls
  Dim x As String
  Dim y As String
  'Dim S As String
  x = ""
  y = "       "
  Print "     ==>TRABAJOS"
  For i = 0 To Problem.NumTrabajos - 1
    y = y & Format(i + 1, "00") & "   "
  Next i
  Print y
  For i = 0 To Problem.NumMaquinas - 1
    For j = 0 To Problem.NumTrabajos - 1
      x = x & Format(Problem.Tiempo(i, j), "00") & "   "
    Next j
    Print "M" & Format(i + 1, "00") & ": " & x
    x = ""
  Next i

  Print ""
   For i = 0 To Problem.NumTrabajos - 1
    For j = 0 To Problem.NumMaquinas - 1
      x = x & Format(Problem.Secuencia(i, j) + 1, "00") & " "
    Next j
    Print "S" & Format(i + 1, "00") & ": " & x
    x = ""
  Next i

End Sub

Private Sub guardaProb_Click() 'código que permite guardar el problema
```

```
dlgCommondialog.CancelError = True
On Error GoTo errhandler
dlgCommondialog.Filter = "Archivo del SEC (*.sec)|*.sec"
dlgCommondialog.DefaultExt = "*.sec"
dlgCommondialog.ShowSave
If dlgCommondialog.FileName <> " " Then
   Guardado = True
   Open dlgCommondialog.FileName For Output As #1
   Print #1, Problem.NumMaquinas 'Guarda el Nro de Trabajos
   Print #1, Problem.NumTrabajos 'Guarda el Nro de Máquinas

   For i = 0 To Problem.NumTrabajos - 1
      For j = 0 To Problem.NumMaquinas - 1
         Print #1, Problem.Secuencia(i, j)
      Next j
   Next i

   For i = 0 To Problem.NumMaquinas - 1
      For j = 0 To Problem.NumTrabajos - 1
            Print #1, Problem.Tiempo(i, j)
      Next j
   Next i
   Close #1
   Me.Caption = "Algoritmos Evolutivo SEC " + dlgCommondialog.FileName
End If
Cls
Exit Sub
errhandler:
   Exit Sub
End Sub

Private Sub leeProb_Click()
   Cls
   Dim LineaDeTexto As String
   With dlgCommondialog
      .InitDir = App.Path
      .DialogTitle = "Abrir"
      .CancelError = False
      .Filter = "Archivos SEC (*.sec)|*.sec"
      .CancelError = True
      On Error GoTo errhandler
      .ShowOpen
      If Len(.FileName) = 0 Then
         Exit Sub
      End If

      sFile = .FileName
      NombreDelProblema = sFile
   End With

   MousePointer = 11
```

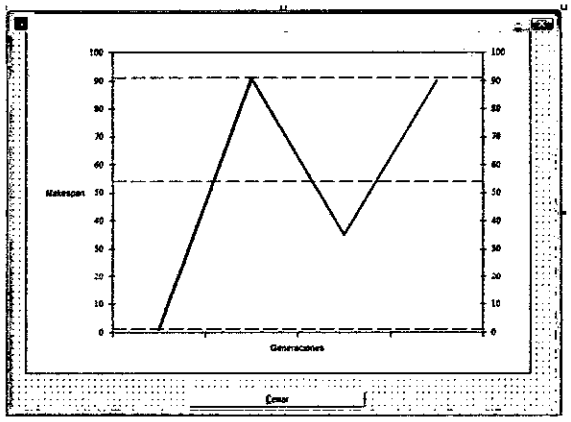
```
  Open dlgCommondialog.FileName For Input As #1
  Line Input #1, LineaDeTexto$
  Problem.NumMaquinas = CInt(LineaDeTexto$)
  Line Input #1, LineaDeTexto$
  Problem.NumTrabajos = CInt(LineaDeTexto$)
  ReDim Problem.Tiempo(Problem.NumMaquinas - 1, Problem.NumTrabajos - 1)
  ReDim Problem.Secuencia(Problem.NumTrabajos - 1, Problem.NumMaquinas - 1)
  ReDim M(Problem.NumMaquinas - 1, Problem.NumTrabajos - 1)

  For i = 0 To Problem.NumTrabajos - 1
    For j = 0 To Problem.NumMaquinas - 1
      Line Input #1, LineaDeTexto$
      Problem.Secuencia(i, j) = CInt(LineaDeTexto$)
    Next j
  Next i

  For i = 0 To Problem.NumMaquinas - 1
    For j = 0 To Problem.NumTrabajos - 1
      Line Input #1, LineaDeTexto$
      Problem.Tiempo(i, j) = CInt(LineaDeTexto$)
    Next j
  Next i

  Close #1
  MousePointer = 0

  Me.Caption = "Algoritmo Evolutivo SEC " + sFile
errhandler:
  Exit Sub

End Sub


Private Sub Command5_Click()
  Dim y As String
  Cls
  y = ""
  GeneraPoblacionInicial
  EvaluaOperacion
  maximoTiempo
  calculaFitness
  For i = 0 To TPo - 1
    For j = 0 To Problem.NumMaquinas * Problem.NumTrabajos - 1

        y = y & ecosistema.poblacion(i).operacion(j)
    Next j
    'Print y & "    " & Format(ecosistema.poblacion(i).makespan, "00") & "    " &
Format(ecosistema.poblacion(i).fitness, "00")
    Print Format(ecosistema.poblacion(i).makespan, "00") & "    " &
Format(ecosistema.poblacion(i).fitness, "00")
    y = ""
```

```
  Next i
End Sub


Private Sub Command6_Click()
  Ruleta
  'Print ""
  'For i = 0 To TPo - 1
  '   For j = 0 To Problem.NumMaquinas * Problem.NumTrabajos - 1

  '         y = y & padres.poblacion(i).operacion(j)
  '   Next j
  '   Print y & "    " & Format(padres.poblacion(i).makespan, "00") & "    " &
Format(padres.poblacion(i).fitness, "00")
  '   y = ""
  'Next i
End Sub


Private Sub Command7_Click()
  'Cruza
  reproduccion
End Sub

Private Sub Command8_Click()
  Dim y As String
  Cls
  y = ""
  For i = 0 To TPo - 1
    For j = 0 To Problem.NumMaquinas * Problem.NumTrabajos - 1
        y = y & ecosistema.poblacion(i).operacion(j) + 1 & "-"
    Next j
    Print y & "    " & Format(ecosistema.poblacion(i).makespan, "00") & "    " &
ecosistema.poblacion(i).fitness
    'Print Format(ecosistema.poblacion(i).makespan, "00") & "    " &
ecosistema.poblacion(i).fitness
    y = ""
  Next i
End Sub



Private Sub Command9_Click()
  Dim NG As Integer
  Cls
  GeneraPoblacionInicial
  NG = InputBox("Ingrese el número de generaciones", "Algoritmo SEC")
  ReDim contieneMak(NG - 1)
  Print "EvaluaOperacion"
  EvaluaOperacion
  Print "maximotiempo"
```

```
    maximoTiempo
    calculaFitness
    For y = 0 To NG - 1
      Print " Seleccionando mejor..."
      mejor
      contieneMak(y) = ecosistema.poblacion(eLite).makespan
      Print " Aplicando Ruleta..."
      Ruleta
      Print " Cruza..."
      reproduccion
      Print "Mutacion..."
      Mutacion
      Print "Evaluando...."
      EvaluaOperacion
      Print "Calculando Fitness..."
      calculaFitness
      Print "Seleccionando el Peor..."
      peOr
      Print "Remplazando..."
      RemplazaPxM
      Print "Generacion " & CStr(y) & " lista"
      If y Mod (6) = 0 Then
        Cls
      End If
    Next y
    MsgBox ("listo")
End Sub


Private Sub SaveIndividuo_Click()
guardaIndividuo
dlgCommondialog.CancelError = True
On Error GoTo errhandler
dlgCommondialog.Filter = "Archivo del SEC (*.sec)|*.sec"
dlgCommondialog.DefaultExt = "*.sec"
dlgCommondialog.ShowSave
If dlgCommondialog.FileName <> " " Then
    Guardado = Ture
    Open dlgCommondialog.FileName For Output As #1
    For i = 0 To Problem.NumMaquinas * Problem.NumTrabajos - 1
     Print #1, Indiv(i)
    Next i
    Close #1
    Me.Caption = "Algoritmos Evolutivo SEC " + dlgCommondialog.FileName
End If
Cls
Exit Sub
errhandler:
  Exit Sub
End Sub
```

```
Private Sub ShowBest_Click()
    For i = 0 To Problem.NumMaquinas * Problem.NumTrabajos - 1
      x = ecosistema.poblacion(eLite).operacion(i) + 1
    Next i
    For i = 0 To Problem.NumMaquinas * Problem.NumTrabajos - 1
      x = ecosistema.poblacion(eLite).operacion(i) + 1
    Next i
End Sub
```

FORMULARIO QUE GRAFICA EL COMPORTAMIENTO DEL ALGORITMO DURANTE SU EJECUCIÓN



CÓDIGO QUE LO RESPALDA EL FORMULARIO:

```
Option Explicit

Private Sub Command1_Click()
  Unload Me
End Sub

Private Sub Form_Load()
  Dim i As Integer
  Dim vardata(1) As Integer
  Me.Caption = "Comportamiento del Algoritmo para: " & NombreDelProblema
  MousePointer = 11
  MSChart1.ChartData = vardata
  MSChart1.ChartData = contieneMak
  For i = 0 To NG - 1
    MSChart1.Column = i + 1
    MSChart1.ColumnLabel = CStr(i)
  Next
  MSChart1.Row = 1
  MSChart1.RowLabel = "Makespan"
  MSChart1.Refresh
  MousePointer = 0
End Sub
```

Anexo N° 2

ANÁLISIS DE VARIANZA DEL COMPORTAMIENTO DEL ALGORITMO EN TODAS LAS INSTANCIAS DE PRUEBA DEL ALGORITMO

Instancia N° 1

Factor TP	Factor NG	REPETICIONES I	II	III	IV	TOTAL	PROMEDIO
30	50	30	30	30	30	120	30.00
30	60	30	30	30	30	120	30.00
30	70	30	30	30	30	120	30.00
30	80	30	30	30	30	120	30.00
40	50	30	30	30	30	120	30.00
40	60	30	30	30	30	120	30.00
40	70	30	30	30	30	120	30.00
40	80	30	30	30	30	120	30.00
50	50	30	30	30	30	120	30.00
50	60	30	30	30	30	120	30.00
50	70	30	30	30	30	120	30.00
50	80	30	30	30	30	120	30.00
60	50	30	30	30	30	120	30.00
60	60	30	30	30	30	120	30.00
60	70	30	30	30	30	120	30.00
60	80	30	30	30	30	120	30.00
	TOTAL	480	480	480	480	1920	30.00

ANVA

FUENTE DE VARIACIÓN	GL	SC	CM	Fc	SIG
TRATAMIENTOS	15	0			
TP	3	0	0		
NG	3	0	0		
TPxGE	9	0	0		
ERROR	48	0	0		
TOTAL	63	0			

CV = 0.00%

Solución óptima: 30

Solución cercana al óptimo: 30

Instancia N° 2

Factor TP	Factor NG	REPETICIONES I	II	III	IV	TOTAL	PROMEDIO
30	50	39	39	39	39	156	39.00
30	60	39	39	39	39	156	39.00
30	70	39	39	39	39	156	39.00
30	80	39	39	39	39	156	39.00
40	50	39	39	39	39	156	39.00
40	60	39	39	39	39	156	39.00
40	70	39	39	39	39	156	39.00
40	80	39	39	39	39	156	39.00
50	50	39	39	39	39	156	39.00
50	60	39	39	39	39	156	39.00
50	70	39	39	39	39	156	39.00
50	80	39	39	39	39	156	39.00
60	50	39	39	39	39	156	39.00
60	60	39	39	39	39	156	39.00
60	70	39	39	39	39	156	39.00
60	80	39	39	39	39	156	39.00
	TOTAL	624	624	624	624	2496	39.00

ANVA

FUENTE DE VARIACIÓN	GL	SC	CM	Fc	SIG
TRATAMIENTOS	15	0			
TP	3	0	0		
NG	3	0	0		
TPxGE	9	0	0		
ERROR	48	0	0		
TOTAL	63	0			

CV = 0.00%

Solución óptima: 39

Solución cercana al óptimo: 39

Instancia N° 3

Factor TP	Factor NG	REPETICIONES I	II	III	IV	TOTAL	PROMEDIO
30	50	34	34	34	34	136	34.00
30	60	34	34	34	34	136	34.00
30	70	34	34	34	34	136	34.00
30	80	34	34	34	34	136	34.00
40	50	34	34	34	34	136	34.00
40	60	34	34	34	34	136	34.00
40	70	34	34	34	34	136	34.00
40	80	34	34	34	34	136	34.00
50	50	34	34	34	34	136	34.00
50	60	34	34	34	34	136	34.00
50	70	34	34	34	34	136	34.00
50	80	34	34	34	34	136	34.00
60	50	34	34	34	34	136	34.00
60	60	34	34	34	34	136	34.00
60	70	34	34	34	34	136	34.00
60	80	34	34	34	34	136	34.00
	TOTAL	544	544	544	544	**2176**	**34.00**

ANVA

FUENTE DE VARIACIÓN	GL	SC	CM	Fc	SIG
TRATAMIENTOS	15	0			
TP	3	0	0		
NG	3	0	0		
TPxGE	9	0	0		
ERROR	48	0	0		
TOTAL	63	0			

CV = 0.00%

Solución óptima: 34
Solución cercana al óptimo: 34

Instancia N° 4

Factor TP	Factor NG	REPETICIONES I	II	III	IV	TOTAL	PROMEDIO
30	50	68	66	68	68	270	67.50
30	60	68	66	68	68	270	67.50
30	70	67	67	67	66	267	66.75
30	80	66	66	67	67	266	66.50
40	50	67	66	67	68	268	67.00
40	60	68	67	68	68	271	67.75
40	70	67	67	66	67	267	66.75
40	80	67	69	66	69	271	67.75
50	50	66	66	67	67	266	66.50
50	60	68	66	67	66	267	66.75
50	70	67	67	68	66	268	67.00
50	80	67	66	67	67	267	66.75
60	50	67	68	66	66	267	66.75
60	60	67	67	67	66	267	66.75
60	70	67	66	68	67	268	67.00
60	80	67	67	67	68	269	67.25
	TOTAL	1074	1067	1074	1074	**4289**	**67.02**

ANVA

FUENTE DE VARIACIÓN	GL	SC	CM	Fc	SIG
TRATAMIENTOS	15	10.23438			
TP	3	2.671875	0.890625	1.3902439	
NG	3	0.921875	0.307292	0.4796748	
TPxGE	9	6.640625	0.737847	1.15176152	
ERROR	48	30.75	0.640625		
TOTAL	63	40.98438			

CV = 1.19%

Solución óptima: 66
Solución con Algoritmo Genético: 67.02

Instancia N° 5

Factor TP	Factor NG	REPETICIONES I	II	III	IV	TOTAL	PROMEDIO
30	50	64	64	64	64	256	64.00
30	60	64	64	64	64	256	64.00
30	70	64	64	64	64	256	64.00
30	80	64	64	64	64	256	64.00
40	50	64	64	64	64	256	64.00
40	60	64	64	64	64	256	64.00
40	70	64	64	64	64	256	64.00
40	80	64	64	64	64	256	64.00
50	50	64	64	64	64	256	64.00
50	60	64	64	64	64	256	64.00
50	70	64	64	64	64	256	64.00
50	80	64	64	64	64	256	64.00
60	50	64	64	64	64	256	64.00
60	60	64	64	64	64	256	64.00
60	70	64	64	64	64	256	64.00
60	80	64	64	64	64	256	64.00
	TOTAL	1024	1024	1024	1024	**4096**	**64.00**

ANVA

FUENTE DE VARIACIÓN	GL	SC	CM	Fc	SIG
TRATAMIENTOS	15	0			
TP	3	0	0		
NG	3	0	0		
TPxGE	9	0	0		
ERROR	48	0	0		
TOTAL	63	0			

CV = 0.00%

Solución óptima: 64
Solución con Algoritmo Genético: 64

Instancia N° 6

Factor TP	Factor NG	REPETICIONES I	II	III	IV	TOTAL	PROMEDIO
30	50	52	53	52	53	210	52.50
30	60	52	52	52	52	208	52.00
30	70	52	53	52	51	208	52.00
30	80	52	52	52	52	208	52.00
40	50	52	52	52	52	208	52.00
40	60	52	52	51	52	207	51.75
40	70	52	52	52	52	208	52.00
40	80	52	52	52	52	208	52.00
50	50	52	52	52	52	208	52.00
50	60	52	52	52	52	208	52.00
50	70	52	52	52	52	208	52.00
50	80	52	52	52	52	208	52.00
60	50	52	52	52	52	208	52.00
60	60	52	52	52	52	208	52.00
60	70	52	52	52	52	208	52.00
60	80	52	52	52	51	207	51.75
	TOTAL	832	834	831	831	**3328**	**52.00**

ANVA

FUENTE DE VARIACIÓN	GL	SC	CM	Fc	SIG
TRATAMIENTOS	15	1.5			
TP	3	0.375	0.12500	1.33333	
NG	3	0.375	0.12500	1.33333	
TPxGE	9	0.750	0.08333	0.88889	
ERROR	48	4.500	0.09375		
TOTAL	63	6.000			

CV = 0.59%

Solución óptima: 51
Solución con Algoritmo Genético: 52

Instancia N° 7

Factor TP	Factor NG	REPETICIONES I	II	III	IV	TOTAL	PROMEDIO
30	50	56	56	56	56	224	56.00
30	60	56	56	56	56	224	56.00
30	70	56	56	56	56	224	56.00
30	80	56	56	56	56	224	56.00
40	50	56	56	56	56	224	56.00
40	60	56	56	56	56	224	56.00
40	70	56	56	56	56	224	56.00
40	80	56	56	56	56	224	56.00
50	50	56	56	56	56	224	56.00
50	60	56	56	56	56	224	56.00
50	70	56	56	56	56	224	56.00
50	80	56	56	56	56	224	56.00
60	50	56	56	56	56	224	56.00
60	60	56	56	56	56	224	56.00
60	70	56	56	56	56	224	56.00
60	80	56	56	56	56	224	56.00
	TOTAL	896	896	896	896	3584	56.00

ANVA

FUENTE DE VARIACIÓN	GL	SC	CM	Fc	SIG
TRATAMIENTOS	15	0			
TP	3	0	0		
NG	3	0	0		
TPxGE	9	0	0		
ERROR	48	0	0		
TOTAL	63	0			

CV = 0.00%

Solución óptima: 56
Solución con Algoritmo Genético: 56

Instancia N° 8

Factor TP	Factor NG	REPETICIONES I	II	III	IV	TOTAL	PROMEDIO
30	50	54	54	54	54	216	54.00
30	60	54	54	54	54	216	54.00
30	70	54	54	54	54	216	54.00
30	80	54	54	54	54	216	54.00
40	50	54	54	54	54	216	54.00
40	60	54	54	54	54	216	54.00
40	70	54	53	54	54	215	53.75
40	80	54	54	54	54	216	54.00
50	50	54	54	54	54	216	54.00
50	60	53	53	54	54	214	53.50
50	70	54	54	54	53	215	53.75
50	80	54	54	54	54	216	54.00
60	50	54	54	54	54	216	54.00
60	60	53	54	54	54	215	53.75
60	70	54	53	54	54	215	53.75
60	80	54	54	54	54	216	54.00
	TOTAL	862	861	864	863	3450	53.91

ANVA

FUENTE DE VARIACIÓN	GL	SC	CM	Fc	SIG
TRATAMIENTOS	15	1.4375			
TP	3	0.3125	0.10417	1.250	
NG	3	0.5625	0.18750	2.250	
TPxGE	9	0.5625	0.06250	0.750	
ERROR	48	4.0000	0.08333		
TOTAL	63	5.4375			

CV = 0.54%

Solución óptima: 53
Solución con Algoritmo Genético: 53.91

Instancia N° 9

Factor TP	Factor NG	REPETICIONES I	II	III	IV	TOTAL	PROMEDIO
30	50	18	18	18	18	72	18.00
30	60	18	18	18	18	72	18.00
30	70	18	18	18	18	72	18.00
30	80	18	18	18	18	72	18.00
40	50	18	18	18	18	72	18.00
40	60	18	18	18	18	72	18.00
40	70	18	18	18	18	72	18.00
40	80	18	18	18	18	72	18.00
50	50	18	18	18	18	72	18.00
50	60	18	18	18	18	72	18.00
50	70	18	18	18	18	72	18.00
50	80	18	18	18	18	72	18.00
60	50	18	18	18	18	72	18.00
60	60	18	18	18	18	72	18.00
60	70	18	18	18	18	72	18.00
60	80	18	18	18	18	72	18.00
	TOTAL	288	288	288	288	1152	18.00

ANVA

FUENTE DE VARIACIÓN	GL	SC	CM	Fc	SIG
TRATAMIENTOS	15	0			
TP	3	0	0		
NG	3	0	0		
TPxGE	9	0	0		
ERROR	48	0	0		
TOTAL	63	0			

CV = 0.00%

Solución óptima: 18
Solución con Algoritmo Genético: 18

Instancia N° 10

Factor TP	Factor NG	REPETICIONES I	II	III	IV	TOTAL	PROMEDIO
30	50	16	16	16	16	64	16.00
30	60	16	16	16	16	64	16.00
30	70	16	16	16	16	64	16.00
30	80	16	16	16	16	64	16.00
40	50	16	16	16	16	64	16.00
40	60	16	16	16	16	64	16.00
40	70	16	16	16	16	64	16.00
40	80	16	16	16	16	64	16.00
50	50	16	16	16	16	64	16.00
50	60	16	16	16	16	64	16.00
50	70	16	16	16	16	64	16.00
50	80	16	16	16	16	64	16.00
60	50	16	16	16	16	64	16.00
60	60	16	16	16	16	64	16.00
60	70	16	16	16	16	64	16.00
60	80	16	16	16	16	64	16.00
	TOTAL	256	256	256	256	1024	16.00

ANVA

FUENTE DE VARIACIÓN	GL	SC	CM	Fc	SIG
TRATAMIENTOS	15	0			
TP	3	0	0		
NG	3	0	0		
TPxGE	9	0	0		
ERROR	48	0	0		
TOTAL	63	0			

CV = 0.00%

Solución óptima: 16
Solución con Algoritmo Genético: 16

Instancia
N° 11

Factor TP	Factor NG	REPETICIONES I	II	III	IV	TOTAL	PROMEDIO
30	50	68	68	68	68	272	68.00
30	60	68	68	68	68	272	68.00
30	70	68	68	68	68	272	68.00
30	80	68	68	68	68	272	68.00
40	50	68	68	68	68	272	68.00
40	60	68	68	68	68	272	68.00
40	70	68	68	68	68	272	68.00
40	80	68	68	68	68	272	68.00
50	50	68	68	68	68	272	68.00
50	60	68	68	68	68	272	68.00
50	70	68	68	68	68	272	68.00
50	80	68	68	68	68	272	68.00
60	50	68	68	68	68	272	68.00
60	60	68	68	68	68	272	68.00
60	70	68	68	68	68	272	68.00
60	80	68	68	68	68	272	68.00
	TOTAL	1088	1088	1088	1088	**4352**	**68.00**

ANVA

FUENTE DE VARIACIÓN	GL	SC	CM	Fc	SIG
TRATAMIENTOS	15	0			
TP	3	0	0		
NG	3	0	0		
TPxGE	9	0	0		
ERROR	48	0	0		
TOTAL	63	0			

CV = 0.00%

Solución óptima: 68

Solución con Algoritmo Genético: 68

ANEXO Nº 3

COMPORTAMIENTO DE LA SOLUCIÓN PARA INSTANCIAS DE LA Nº12 A LA Nº30

Instancia	Nombre	Tamaño	Makespan	Tiempo (seg)
12	JSSP8T9M	8X9	78	11
13	JSSP13T15M	13X15	124	21
14	JSSP10T12M	10X12	99	14
15	JSSP15T20M	15X20	163	35
16	JSSP10T10M	10X10	91	11
17	JSSP11T12M	11X12	108	16
18	JSSP20T15M	20X15	148	35
19	JSSP13T13M	13X13	118	20
20	JSSP15T15M	15X15	153	28
21	JSSP9T13M	9X13	98	14
22	JSSP11T12M	11X12	102	16
23	JSSP9T7M	9X7	69	8
24	JSSP14T14M	14X14	124	26
25	JSSP17T17M	17X17	156	32
26	JSSP15T8M	15X8	104	16
27	JSSP7T14M	7X14	84	14
28	JSSP20T20M	20X20	188	55
29	JSSP18T14M	18X14	156	33
30	JSSP11T11M	11X11	96	15

ANEXO N° 4

APLICACIÓN DEL ALGORITMO A UN EJEMPLO REAL

FACTORIA XYZ S.C.R.L

Es una empresa fundada el dd/mm/aaaa, con número de RUC xxxxxxxxxxx. Está ubicada en la calle1 N° xxx (altura de la cuadra xx de Av. yyyyyyyy), del AA.HH xxxxxxxxxxxxxx del distrito de xxxxxxxxx del Departamento de Piura, Región Grau. Teléfono xxxxxx.

Es una empresa dedicada principalmente a la fundición y soldadura de aluminio utilizada en la reparación de monoblocks, culatas, cigüeñales, etc. de vehículos de transporte liviano y pesado. A tal efecto cuenta con maquinaria y equipo especializado para realizar las tareas rutinarias del día; así mismo cuenta con 22 trabajadores especializados, cuatro de ellos son hermanos y dueños de la factoría.

Las operaciones diariamente en la factoría se inician con la recepción de los trabajos que llegan al taller los que son atendidos por cualquiera de los hermanos quien se encarga de distribuir el trabajo entre los 17 trabajadores restantes de acuerdo a lo solicitado por el cliente y de la especialidad de cada trabajador, este encargado de distribuir el trabajo estima el tiempo que demorará el trabajo hasta concluirlo y normalmente informa al cliente el tiempo estimado de terminación del trabajo solicitado que generalmente es al siguiente día o después de dos días.

Como se puedo observar no están estandarizados los procedimientos de recepción y entrega del trabajo terminado al cliente, como tampoco están estandarizados los tiempos de maquinado en las máquinas de las tareas que requiere cada trabajo; sin embrago por el tiempo que llevan trabajando en la misma actividad conocen el tiempo medio de maquinado de cada tarea en cada una de las máquinas y la secuencia de tareas que requiere cada trabajo solicitado.

En estas circunstancias se presenta el siguiente ejemplo de un día de trabajo en la factoría y el tiempo total que hubiera demandado realizar los trabajos utilizando el algoritmo genético propuesto en este proyecto.

Máquinas Utilizadas en los trabajos solicitados

M1: Rectificadora mecánica de cilindros (120 minutos/ 4 cilindros)

M2: Rectificadora digital de cilindros (80 minutos/4 cilindros)

M3: Máquina pulidora de cilindros (30 minutos/4 cilindros)

M4: Rectificadora de cigüeñal (cigüeñal grande 80 cm. 120 minutos, 40 cm. 60 minutos))

M5: Rectificadora de superficies planas (20 minutos)

M6: Soldadora eléctrica

M7: Sujeción en banco de relleno y fundición de aluminio culata de auto (30 minutos)

M8: Banco de asentado de válvulas, colocación de guías de válvula y asentar árboles de

levas de Culata de auto (90 minutos)

M9: Taladrar huecos, pases de agua (120 minutos)

M10: Soldadora oxiacetilénica.

M11: Taladro de banco

M12: Torno horizontal

Trabajos solicitados por clientes un día particular (Sábado 27/12/2014)

T1: Rectificar Cilindro moto CB125:

Requiere posicionar cilindro en rectificadora mecánica de cilindros, maquinarlo y retirar cilindro, tiempo de maquinado 30 minutos, luego pasa a la máquina pulidora de cilindros (10 minutos).

T2: Rellenar y rectificar monoblock de auto:

Requiere de fundir material en monoblock (30 min), esperar enfriar por 30 minutos, luego posicionar y maquinar monoblock en Rectificadora de superficies planas 20 minutos, posteriormente requiere rectificar cilindros en rectificadora mecánica 120 minutos y finalmente pulir superficies de cilindros en pulidora de cilindros 30 minutos.

T3: Rellenar y rectificar culata de auto:

Requiere fundir aluminio en culata (30 minutos) esperar enfriar (30 minutos) y maquinar en rectificadora de superficies planas (20 minutos); posteriormente requiere taladrar huecos y pases de agua (120 minutos)

T4: Soldar parte de auto (10 min)

T5: Soldar autoparte (20 min.)

T6: Rectificar Cilindros motor de automóvil:

Requiere posicionar y rectificar cilindros en rectificadora mecánica 120 minutos y pulir superficies de cilindros en pulidora de cilindros 30 minutos.

T7: Hacer pieza según diseño:

Requiere soldadora eléctrica para soldar partes en base (20 minutos) rectificar soldadura en torno (20 minutos) luego requiere taladrar agujeros (20 minutos)

T8: Taladrar superficie

Requiere 5 agujeros de ¾" (20 minutos)

Secuencia de trabajos

T1: M1 – M3

T2: M7 – enfriar – M5 – M1 – M3

T3: M7 – enfriar – M5 – M9

T4: M6

T5: M10

T6: M1 – M3

T7: M6 – M12 – M11

T8: M11

Tiempo de procesamiento de los trabajos en las máquinas:

	T1	T2	T3	T4	T5	T6	T7	T8
M1	30	120	0	0	0	120	0	0
M2	0	0	0	0	0	0	0	0
M3	10	30	0	0	0	30	0	0
M4	0	0	0	0	0	0	0	0
M5	0	20	20	0	0	0	0	0
M6	0	0	0	10	0	0	20	0
M7	0	30	30	0	0	0	0	0
M8	0	0	0	0	0	0	0	0
M9	0	0	90	0	0	0	0	0
M10	0	0	0	0	20	0	0	0
M11	0	0	0	0	0	0	20	20
M12	0	0	0	0	0	0	20	0
M13	0	30	0	0	0	0	0	0
M14	0	0	30	0	0	0	0	0

El tiempo de enfriado de 30 minutos, se representa como una máquina que comprende un ventilador funcionando por 30 minutos, de no hacerlo de esa manera se tendría que esperar 120 minutos de enfriamiento a la intemperie; esta situación se representa en los tiempos de procesamiento como M13 y M14.

Comparación de solución óptima con Algoritmo genético

Solución Proporcionada por el Algoritmo genético

M01: 6 6 6 6 6 6 6 6 6 6 6 6 2 2 2 2 2 2 2 2 2 2 2 1 1 1 0
M03: 0 0 0 0 0 0 0 0 0 0 0 0 6 6 6 0 0 0 0 0 0 0 0 2 2 2 1
M05: 0 0 0 0 0 0 3 3 0 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
M06: 4 7 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
M07: 3 3 3 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
M09: 0 0 0 0 0 0 0 0 3 3 3 3 3 3 3 3 0 0 0 0 0 0 0 0 0 0 0
M10: 5 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
M11: 8 8 0 0 0 7 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
M12: 0 0 0 7 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
M13: 0 0 0 0 0 0 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
M14: 0 0 0 3 3 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Makespan:28

Equivale a realizar todos los trabajos en 280 minutos (4.67 hrs)
Las máquinas M13 y M14, es el estado de esperar enfriar.
Cada unidad de tiempo es de 10 minutos.

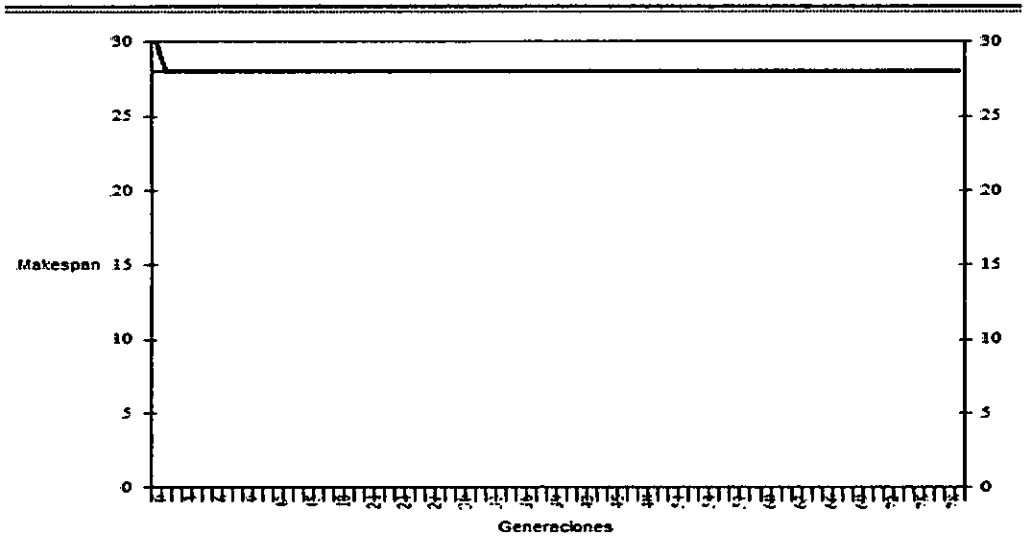
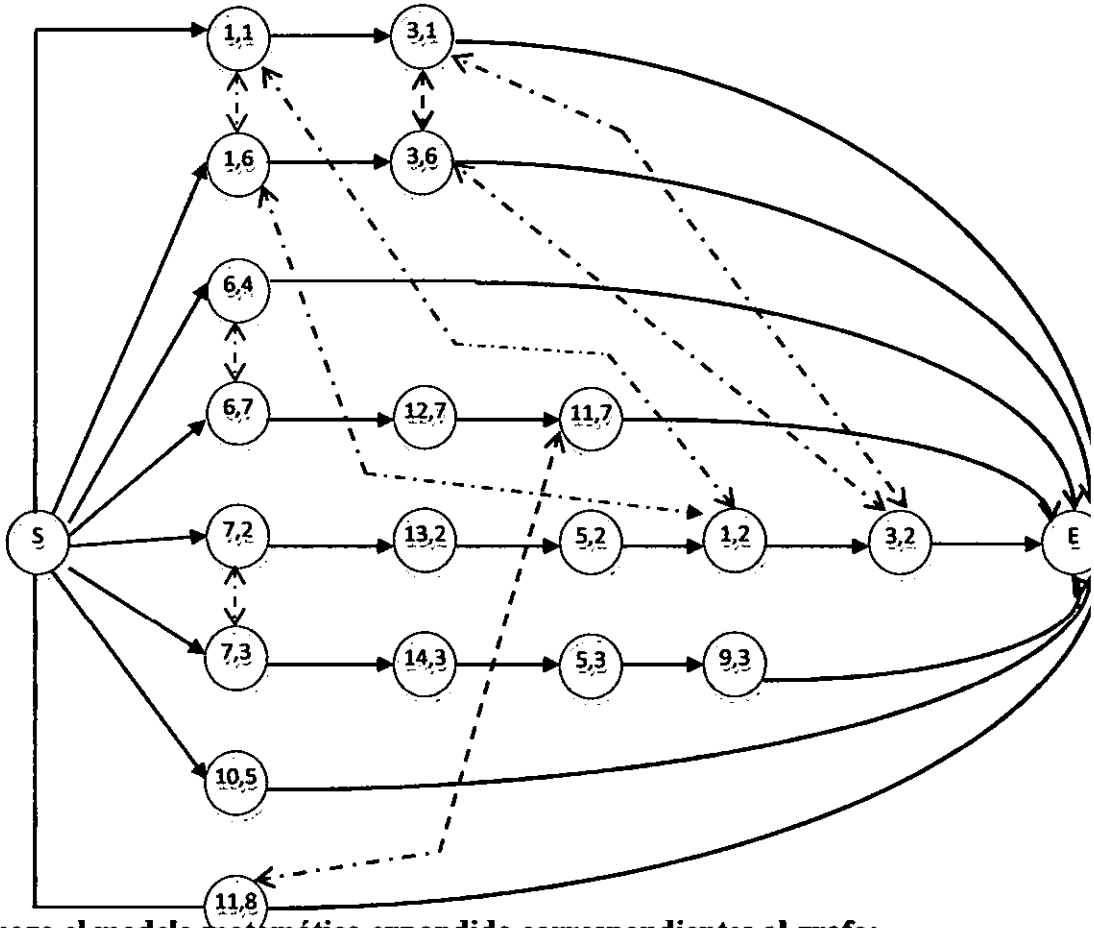


Figura Gantt: Comportamiento del algoritmo genético

Solución optima que proporciona el modelo matemático, utilizando WinQSB

En primer lugar se muestra el grafo disyuntivo correspondiente:



Luego el modelo matemático expandido correspondientes al grafo:

M=14 máquinas
N=8 trabajos

N° de variables = 19 + 9 + 1 = 29 (19 variables enteras de tiempos de inicio
de cada Operación 9 variables
binarias, 1 variable Cmax)

N° de restricciones = 11 de precedencia
19 en cada nodo
18 de disyunción
48

Min Z = Cmax

s.a

1. $Y_{3,1} - Y_{1,1} >= d_{1,1}$
2. $Y_{3,6} - Y_{1,6} >= d_{1,6}$
3. $Y_{12,7} - Y6,7 >= d_{6,7}$
4. $Y_{11,7} - Y_{12,7} >= d_{12,7}$
5. $Y_{13,2} - Y_{7,2} >= d_{7,2}$
6. $Y_{5,2} - Y_{13,2} >= d_{13,2}$
7. $Y_{1,2} - Y_{5,2} >= d_{5,2}$
8. $Y_{3,2} - Y_{1,2} >= d_{1,2}$
9. $Y_{14,3} - Y_{7,3} >= d_{7,3}$
10. $Y_{5,3} - Y_{14,3} >= d_{14,3}$
11. $Y_{9,3} - Y_{5,3} >= d_{5,3}$

12. Cmax $- Y_{3,1} >= d_{3,1}$
13. Cmax $- Y_{1,1} >= d_{1,1}$
14. Cmax $- Y_{3,6} >= d_{3,6}$
15. Cmax $- Y_{1,6} >= d_{1,6}$
16. Cmax $- Y_{6,4} >= d_{6,4}$
17. Cmax $- Y_{11,7} >= d_{11,7}$
18. Cmax $- Y_{12,7} >= d_{12,7}$
19. Cmax $- Y_{6,7} >= d_{6,7}$

20. Cmax - $Y_{3,2}$ >= $d_{3,2}$

21. Cmax - $Y_{1,2}$ >= $d_{1,2}$

22. Cmax - $Y_{5,2}$ >= $d_{5,2}$

23. Cmax - $Y_{13,2}$ >= $d_{1,3}$

24. Cmax - $Y_{7,2}$ >= $d_{7,2}$

25. Cmax - $Y_{9,3}$ >= $d_{9,3}$

26. Cmax - $Y_{5,3}$ >= $d_{5,3}$

27. Cmax - $Y_{14,3}$ >= $d_{14,3}$

28. Cmax - $Y_{7,3}$ >= $d_{7,3}$

29. Cmax - $Y_{10,5}$ >= $d_{10,5}$

30. Cmax - $Y_{11,8}$ >= $d_{11,5}$

31. $Y_{1,6}$ – $Y_{1,1}$ >= $d_{1,1}$ ó $Y_{1,1}$ – $Y_{1,6}$ >= $d_{1,6}$

32. $Y_{6,7}$ – $Y_{6,4}$ >= $d_{6,4}$ ó $Y_{6,4}$ – $Y_{6,7}$ >= $d_{6,7}$

33. $Y_{7,3}$ – $Y_{7,2}$ >= $d_{7,2}$ ó $Y_{7,2}$ – $Y_{7,3}$ >= $d_{7,3}$

34. $Y_{11,7}$ – $Y_{11,8}$ >= $d_{11,8}$ ó $Y_{11,8}$ – $Y_{11,7}$ >= $d_{11,7}$

35. $Y_{1,6}$ – $Y_{1,2}$ >= $d_{1,2}$ ó $Y_{1,2}$ – $Y_{1,6}$ >= $d_{1,6}$

36. $Y_{1,2}$ – $Y_{1,1}$ >= $d_{1,1}$ ó $Y_{1,1}$ – $Y_{1,2}$ >= $d_{1,2}$

37. $Y_{3,1}$ – $Y_{3,6}$ >= $d_{3,6}$ ó $Y_{3,6}$ – $Y_{3,1}$ >= $d_{3,1}$

38. $Y_{3,2}$ – $Y_{3,1}$ >= $d_{3,1}$ ó $Y_{3,1}$ – $Y_{3,2}$ >= $d_{3,2}$

39. $Y_{3,2}$ – $Y_{3,6}$ >= $d_{3,6}$ ó $Y_{3,6}$ – $Y_{3,2}$ >= $d_{3,2}$

Las nueve (9) restricciones anteriores (de la 31 a la 39), se convierten en 18 restricciones

1. $Y_{1,6}$ – $Y_{1,1}$ + M(1 – Y1) >= $d_{1,1}$

2. $Y_{1,1}$ – $Y_{1,6}$ + M(Y1) >= $d_{1,6}$

3. $Y_{6,7}$ – $Y_{6,4}$ + M(1 – Y2) >= $d_{6,4}$

4. $Y_{6,4}$ – $Y_{6,7}$ + M(Y2) >= $d_{6,7}$

5. $Y_{7,3}$ – $Y_{7,2}$ + M(1 – Y3) >= $d_{7,2}$

6. $Y_{7,2}$ – $Y_{7,3}$ + M(Y3) >= $d_{7,3}$

7. $Y_{11,7}$ – $Y_{11,8}$ + M(1 – Y4) >= $d_{11,8}$

8. $Y_{11,8}$ – $Y_{11,7}$ + M(Y4) >= $d_{11,7}$

9. $Y_{1,6}$ – $Y_{1,2}$ + M(1 – Y5) >= $d_{1,2}$

10. $Y_{1,2}$ – $Y_{1,6}$ + M(Y5) >= $d_{1,6}$

11. $Y_{1,2}$ – $Y_{1,1}$ + M(1 – Y6) >= $d_{1,1}$

12. $Y_{1,1}$ – $Y_{1,2}$ + M(Y6) >= $d_{1,2}$

13. $Y_{3,1}$ – $Y_{3,6}$ + M(1 – Y7) >= $d_{3,6}$

14. $Y_{3,6} - Y_{3,1} + M(Y7) >= d_{3,1}$

15. $Y_{3,2} - Y_{3,1} + M(1 - Y8) >= d_{3,1}$

16. $Y_{3,1} - Y_{3,2} + M(Y8) >= d_{3,2}$

17. $Y_{3,2} - Y_{3,6} + M(1 - Y9) >= d_{3,6}$

18. $Y_{3,6} - Y_{3,2} + M(Y9) >= d_{3,2}$

Yij >= 0 y enteros, Yk = binarias (0, 1); k= 1, 2, ... 9

El modelo anterior proporciona la siguiente solución:

	20:40:15		Sunday	December	28	2014
	Decision Variable	Solution Value	Unit Cost or Profit c(j)	Total Contribution	Reduced Cost	Basis Status
1	Y3_1	270.0000	0	0	0	basic
2	Y1_1	240.0000	0	0	0	basic
3	Y3_6	210.0000	0	0	0	basic
4	Y1_6	0	0	0	1.0000	at bound
5	Y6_4	240.0000	0	0	0	basic
6	Y6_7	220.0000	0	0	0	basic
7	Y11_7	260.0000	0	0	0	basic
8	Y12_7	240.0000	0	0	0	basic
9	Y13_2	70.0000	0	0	0	basic
10	Y7_2	40.0000	0	0	0	basic
11	Y5_2	100.0000	0	0	0	basic
12	Y3_2	240.0000	0	0	0	basic
13	Y9_3	190.0000	0	0	0	basic
14	Y11_8	240.0000	0	0	0	basic
15	Y10_5	260.0000	0	0	0	basic
16	Y1_2	120.0000	0	0	0	basic
17	Y14_3	140.0000	0	0	0	basic
18	Y7_3	70.0000	0	0	0	basic
19	Y5_3	170.0000	0	0	0	basic
20	Cmax	280.0000	1.0000	280.0000	0	basic
21	Y1	0	0	0	0	at bound
22	Y2	0	0	0	0	at bound
23	Y3	1.0000	0	0	0	at bound
24	Y4	1.0000	0	0	0	at bound
25	Y5	0	0	0	0	at bound
26	Y6	0	0	0	0	at bound
27	Y7	1.0000	0	0	0	at bound
28	Y8	0	0	0	0	at bound
29	Y9	1.0000	0	0	0	at bound
	Objective	Function	(Min.) =	280.0000		

Esta solución es coincidente con la que proporciona el algoritmo genético construido.

WEB15X15-2 **AG**

Nb of jobs	: 15	15
Nb of Machines	: 15	15
Time seed	: 1314640371	80seg
Machine seed	: 386720536	Core 2 duo
Makespan	: 1244	1426 14.63%

Times

86	60	10	59	65	94	71	25	98	49	43	8	90	21	73
68	28	38	36	93	35	37	28	62	86	65	11	20	82	23
33	67	96	91	83	81	60	88	20	62	22	79	38	40	82
13	14	73	88	24	16	78	70	53	68	73	90	58	7	4
93	52	63	13	19	41	71	59	19	60	85	99	73	95	19
62	60	93	16	10	72	88	69	58	41	46	63	76	83	62
50	68	90	34	44	5	8	25	70	53	78	92	62	85	70
60	64	92	44	63	91	21	1	96	19	59	12	41	11	94
93	46	51	37	91	90	63	40	68	13	16	83	49	24	23
5	35	21	14	66	3	6	98	63	64	76	94	17	62	37
35	42	62	68	73	27	52	39	41	25	9	34	50	41	98
23	32	35	10	29	68	20	8	58	62	39	32	8	33	91
28	31	3	28	66	59	24	45	81	8	44	42	2	23	53
11	93	27	59	62	23	23	7	77	64	60	97	36	53	72
36	98	38	24	84	47	72	1	91	85	68	42	20	30	30

Machines

10	15	5	14	11	4	8	9	1	6	2	3	13	7	12
11	9	12	15	4	14	10	8	5	3	7	2	6	13	1
8	1	7	6	15	14	3	12	5	13	2	10	4	11	9
10	12	15	1	2	9	6	11	13	5	14	4	7	8	3
12	5	14	4	9	2	11	13	3	15	7	8	1	10	6
6	3	2	11	1	5	9	15	7	4	10	8	12	13	14
6	11	14	1	10	9	2	12	15	8	13	3	7	5	4
13	1	10	4	14	7	6	8	3	15	12	9	11	2	5
12	11	6	14	2	10	9	8	4	7	1	3	15	13	5
3	15	4	11	7	2	1	14	12	5	6	9	8	13	10
12	15	14	6	5	10	2	7	13	1	3	9	11	4	8
13	4	11	9	5	8	14	12	15	2	3	1	6	7	10
9	14	6	1	12	10	5	13	2	11	7	3	8	15	4
3	6	5	4	10	2	12	14	8	7	11	15	1	9	13
2	11	5	3	1	8	7	10	12	13	6	15	4	14	9

WEB15X15-3 **AG**

Nb of jobs	: 15	15
Nb of Machines	: 15	15
Time seed	: 1227221349	80seg
Machine seed	: 316176388	Core 2 duo
Makespan	: 1222	1398 14.40%

Times

69	81	81	62	80	3	38	62	54	66	88	82	3	12	88
83	51	47	15	89	76	52	18	22	85	26	30	5	89	22
62	47	93	54	38	78	71	96	19	33	44	71	90	9	21
33	82	80	30	96	31	11	26	41	55	12	10	92	3	75
36	49	10	43	69	72	19	65	37	57	32	11	73	89	12
83	32	6	13	87	94	36	76	46	30	56	62	32	52	72
29	78	21	27	17	43	14	15	16	49	72	19	99	38	64
12	74	4	3	15	62	50	38	49	25	18	55	5	71	27
69	13	33	47	86	31	97	48	25	40	94	22	61	59	16
27	4	35	80	49	46	84	46	96	72	18	23	96	74	23
36	17	81	67	47	5	51	23	82	35	96	7	54	92	38
78	58	62	43	1	56	76	49	80	26	79	9	24	24	42
38	86	38	38	83	36	11	17	99	14	57	64	58	96	17
10	86	93	63	61	62	75	90	40	77	8	27	96	69	64
73	12	14	71	3	47	84	84	53	58	95	87	90	68	75

Machines

8	12	9	4	13	2	14	1	15	7	10	5	3	11	6
13	2	12	10	7	4	3	5	6	9	14	15	11	1	8
2	3	10	1	4	6	9	5	15	11	13	14	8	7	12
14	11	7	3	15	8	5	12	1	6	10	4	9	2	13
2	9	5	15	7	6	4	3	10	11	14	8	12	1	13
6	15	3	13	11	2	12	5	7	10	1	14	9	4	8
6	3	1	2	9	15	12	11	8	10	7	13	5	14	4
5	8	11	2	10	9	3	15	12	4	6	7	14	13	1
15	12	1	10	11	6	4	13	9	14	7	2	8	3	5
10	1	4	11	13	14	6	2	7	15	9	12	3	8	5
8	3	2	13	4	15	5	7	6	10	9	14	11	1	12
1	9	15	13	10	6	7	11	8	12	4	5	2	14	3
9	13	11	12	15	4	7	2	5	6	1	10	14	3	8
14	3	12	1	15	11	4	2	13	5	6	7	8	10	9
2	14	1	12	3	11	5	9	4	6	8	7	10	13	15

WEB15X15-4 **AG**

Nb of jobs	: 15	15
Nb of Machines	: 15	15
Time seed	: 342269428	80seg
Machine seed	: 1806358582	Core 2 duo
Makespan	: 1181	1368 15.83%

Times

72	51	42	31	61	46	88	33	27	85	70	56	70	50	25
19	79	79	47	40	67	43	65	84	61	30	56	19	91	68
94	7	2	95	60	82	76	36	8	85	7	44	2	72	91
58	67	84	34	19	19	94	41	98	96	25	40	74	88	74
45	60	8	29	32	42	25	4	71	79	93	28	30	17	43
84	56	46	93	66	84	40	4	15	15	54	39	77	55	31
65	91	17	47	77	68	62	22	72	47	38	7	11	22	63
12	21	60	42	22	84	60	52	25	53	53	56	29	83	32
48	28	70	26	68	4	19	92	24	54	57	47	84	85	95
36	34	65	64	30	41	53	74	44	13	41	6	32	94	37
62	9	89	37	28	23	13	60	46	94	85	72	18	79	11
74	61	43	26	97	62	40	60	62	78	42	8	21	11	70
9	22	9	8	54	32	92	76	2	63	63	98	42	12	41
67	7	91	52	87	4	1	56	82	47	35	8	92	39	11
44	24	24	14	34	57	30	64	4	14	69	95	22	60	61

Machines

4	8	7	15	10	9	6	5	11	2	1	13	12	3	14
2	12	1	6	9	14	4	11	10	3	13	7	5	8	15
8	4	9	12	1	5	10	14	2	11	7	6	15	13	3
7	12	6	14	4	3	2	5	10	13	9	1	15	8	11
2	12	7	6	9	8	13	10	3	15	14	4	1	5	11
12	3	13	2	15	1	7	10	4	14	8	11	5	6	9
15	8	12	1	13	9	10	11	4	14	5	2	3	7	6
8	5	14	6	15	12	3	13	7	2	1	11	10	4	9
8	5	15	11	4	1	14	9	2	3	7	13	12	10	6
3	13	11	4	7	6	15	10	14	5	1	12	8	9	2
12	10	4	8	7	5	1	3	6	2	11	9	14	13	15
1	5	2	10	3	12	14	11	8	7	9	13	15	4	6
7	1	5	11	3	15	10	14	12	13	2	9	6	4	8
6	11	14	3	8	5	9	7	10	1	13	12	2	15	4
12	3	1	5	15	6	10	11	7	13	9	2	8	4	14

WEB15X15-5 **AG**

Nb of jobs	: 15	15
Nb of Machines	: 15	15
Time seed	: 1603221416	80seg
Machine seed	: 1501949241	Core 2 duo
Makespan	: 1233	1451 17.68%

Times

40	96	59	95	76	75	23	65	65	16	71	52	84	99	24
2	88	99	52	68	13	38	35	57	37	93	38	68	94	71
87	46	14	87	30	79	62	37	54	1	97	16	2	51	96
19	15	42	8	72	15	76	25	78	84	62	70	81	16	97
68	71	3	68	91	37	73	21	85	79	51	50	21	30	64
14	1	29	72	6	31	98	50	83	2	86	33	33	98	59
21	80	99	70	80	71	47	96	56	78	53	10	92	1	33
29	85	89	10	30	38	38	48	16	65	90	73	88	46	47
37	9	49	23	1	78	39	15	9	41	35	83	8	61	60
1	73	47	46	10	37	60	84	26	11	37	79	75	49	51
22	49	33	2	24	3	73	68	21	61	69	94	43	39	48
81	46	21	23	86	19	64	52	22	50	11	73	77	16	75
21	80	30	32	22	23	85	92	14	13	68	60	45	32	90
29	95	52	59	33	12	73	96	75	12	83	3	90	57	6
94	18	54	42	70	29	43	50	75	70	40	48	1	27	12

Machines

13	2	5	10	14	1	12	9	4	6	7	15	11	3	8
6	2	15	11	14	10	7	13	9	3	8	1	4	5	12
7	4	11	8	14	5	6	10	9	3	1	2	12	15	13
11	8	6	1	10	14	3	9	2	15	12	4	13	7	5
7	13	15	2	8	6	12	1	3	4	9	14	5	10	11
5	8	7	1	9	14	13	15	4	3	6	10	11	2	12
11	12	7	10	1	3	2	9	13	8	6	4	14	15	5
4	11	6	7	9	5	1	15	3	8	10	12	13	2	14
2	3	7	8	11	13	15	9	1	4	14	6	5	10	12
13	8	7	15	4	5	1	14	11	9	12	10	6	3	2
5	4	9	15	3	14	6	7	11	13	8	12	2	10	1
7	13	8	6	3	5	14	12	9	1	11	4	2	10	15
13	14	9	8	2	7	1	6	10	11	5	3	15	4	12
13	2	5	9	7	11	8	4	1	6	14	3	10	15	12
8	6	1	11	3	4	10	7	12	9	2	5	15	13	14

WEB15X15-6 **AG**

Nb of jobs	: 15	15
Nb of Machines	: 15	15
Time seed	: 1357584978	80seg
Machine seed	: 1734077082	Core 2 duo
Makespan	: 1243	1402 12.79%

Times

96	23	71	26	28	16	27	71	18	57	43	5	12	91	63
32	81	95	79	55	45	60	73	23	44	92	20	5	72	73
63	93	63	79	10	66	27	93	24	26	8	69	29	66	97
80	87	68	23	54	16	68	32	74	3	2	71	4	67	28
46	96	11	41	93	2	98	10	43	65	27	57	75	87	81
5	91	92	87	66	36	67	88	92	27	13	7	95	66	13
90	33	78	76	93	67	82	94	12	5	85	42	4	2	70
79	24	41	83	45	29	3	42	5	44	83	59	60	78	44
19	55	20	74	66	37	55	63	40	73	55	84	54	62	6
27	59	6	90	6	37	64	35	25	59	77	30	1	7	70
4	53	6	10	51	89	38	38	35	44	99	88	52	16	99
28	11	76	51	35	60	44	39	66	49	40	34	80	38	29
31	32	40	25	40	85	39	61	15	41	93	64	16	81	97
9	21	8	55	79	76	79	61	68	99	24	23	92	91	22
80	30	67	58	45	29	48	28	64	63	80	23	93	55	48

Machines

8	13	6	9	4	15	14	10	1	2	5	3	7	12	11
9	1	6	14	7	2	5	3	4	12	11	13	10	15	8
7	9	8	2	10	6	13	5	1	3	15	12	4	14	11
13	1	15	8	14	2	7	9	12	3	4	10	6	11	5
9	3	13	12	11	10	5	4	15	6	1	8	7	14	2
14	9	7	12	15	10	13	8	1	2	11	3	6	4	5
5	2	3	14	8	7	1	9	13	15	11	10	6	4	12
5	11	15	4	7	14	12	10	1	6	2	13	9	8	3
8	2	6	5	15	9	1	10	13	4	11	3	14	12	7
4	7	14	5	6	11	3	9	13	12	10	8	15	2	1
2	5	8	3	15	11	13	14	7	6	10	1	4	9	12
1	8	15	14	9	4	7	10	6	13	11	5	2	3	12
15	11	1	9	13	14	5	12	2	4	7	6	3	10	8
14	5	9	7	6	3	10	1	12	13	2	4	15	8	11
15	3	7	5	4	12	2	6	8	1	9	11	10	13	14